2013-1489

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

VIRNETX, INC.,

Plaintiff-Appellee,

and

SCIENCE APPLICATIONS INTERNATIONAL CORPORATION,

Plaintiff-Appellee,

v.

CISCO SYSTEMS, INC.,

Defendant,

and

APPLE INC.,

Defendant-Appellant.

Appeal from the United States District Court for the Eastern District of Texas in case no. 10-CV-0417, Chief Judge Leonard Davis.

NON-CONFIDENTIAL BRIEF FOR DEFENDANT-APPELLANT APPLE INC.

JONATHAN G. CEDARBAUM
BRITTANY BLUEITT AMADI
LEAH LITMAN
WILMER CUTLER PICKERING
HALE AND DORR LLP
1875 Pennsylvania Ave., NW
Washington, DC 20006
(202) 663-6000

DANNY L. WILLIAMS
WILLIAMS, MORGAN & AMERSON, P.C.
10333 Richmond, Suite 1100
Houston, TX 77042
(713) 934-7000

WILLIAM F. LEE
MARK C. FLEMING
LAUREN B. FLETCHER
REBECCA BACT
WILMER CUTLER PICKERING
HALE AND DORR LLP
60 State Street
Boston, MA 02109
(617) 526-6000

Attorneys for Defendant-Appellant Apple Inc.

October 17, 2013

CERTIFICATE OF INTEREST

Pursuant to Federal Circuit Rule 47.4, counsel of record for Defendant-Appellant Apple Inc. certify as follows:

1. The full name of every party represented by us is:
   Apple Inc.

2. The names of the real parties in interest represented by us are:
   Apple Inc.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the parties represented by us are:
   None

4. The names of all law firms and the partners or associates that appeared for the parties represented by us in the trial court, or are expected to appear in this Court, are:

   ALBRITTON LAW FIRM: Eric M. Albritton

   FINDLAY CRAFT: Roger Brian Craft, Eric Hugh Findlay

   KENYON & KENYON: Megan Whyman Olesek, Marcia H. Sundeen

   LAW OFFICE OF SCOTT WOLOSON: Scott Edward Woloson

   WILLIAMS, MORGAN & AMERSON, P.C.: Ruben Singh Bains, Christopher Needham Cravey, Kyung Kim, Terry D. Morgan, Leisa Talbert Peschel, Matthew Richard Rodgers, Danny Lloyd Williams,

   WILMER CUTLER PICKERING HALE AND DORR LLP: Brittany Blueitt Amadi, Rebecca Bact, Jonathan G. Cedarbaum, Mark C. Fleming, Lauren B. Fletcher, William F. Lee, Leah Litman

   WONG, CABELLO, LUTSCH, RUTHERFORD & BRUCCULERI: Stephen Edwards

Dated: October 17, 2013

/s/ William F. Lee
WILLIAM F. LEE

TABLE OF CONTENTS

Page(s)

CONFIDENTIAL MATERIAL OMITTED

The material omitted on pages 18 and 19, in the fourth and fifth lines of page 59, and in footnote 13 on page 59 describes the results of internal Apple customer surveys. This material was designated confidential by Appellant Apple Inc. pursuant to the Protective Order entered August 2, 2011 and amended March 5, 2012.

TABLE OF AUTHORITIES

CASES

STATUTES

OTHER AUTHORITIES

STATEMENT OF RELATED CASES

No appeal in this case was previously before this Court or any other court. The en banc Court is currently considering whether to defer to a district court's claim construction. *Lighting Ballast Control LLC v. Philips Elec. N. Am. Corp.*, 500 F. App'x 951 (Fed. Cir. 2013). Petitions for rehearing en banc are also pending in *Commil USA, LLC v. Cisco Systems, Inc.*, 720 F.3d 1361 (Fed. Cir. 2013), on which Defendant-Appellant Apple Inc. ("Apple") relies. An appeal from the judgment in *Suffolk Technologies LLC v. AOL Inc.*, 2013 U.S. Dist. LEXIS 64630 (E.D. Va. Apr. 12, 2013), in which a district court excluded testimony by the plaintiff's damages expert Roy Weinstein under a "Nash bargaining solution" theory—the very same expert and theory proffered by Plaintiff-Appellee VirnetX, Inc. ("VirnetX") here—is pending before this Court as No. 2013-1392. Counsel for Apple are aware of no other case pending in this Court that would directly affect or be directly affected by the Court's decision in this appeal.

Four cases pending before the United States District Court for the Eastern District of Texas may be affected by the Court's decision in this appeal. *First*, the district court severed into a separate case VirnetX's request for an ongoing royalty from Apple in this action. *See VirnetX, Inc. v. Apple Inc.*, No. 13-cv-211 (E.D. Tex.); A75. *Second*, on the same day as the verdict in this case, VirnetX filed a separate lawsuit against Apple for infringement of all four patents at issue in this appeal (U.S. Patent Nos. 6,502,135 ("the '135 patent"); 7,490,151 ("the '151 patent"); 7,418,504 ("the '504 patent"); and 7,921,211 ("the '211 patent")). *See VirnetX, Inc. v. Apple Inc.*, No. 12-cv-855 (E.D. Tex.). *Third*, VirnetX's claims against Defendant Cisco Systems, Inc., which involve three of the same patents at issue in this appeal (the '135, '211, and '504 patents), also remain pending before the district court. *See VirnetX, Inc. v. Cisco Sys., Inc.*, No. 10-cv-417 (E.D. Tex.). *Finally*, VirnetX has filed suit against Microsoft, alleging infringement of all four patents at issue in this appeal. *See VirnetX, Inc. v. Microsoft Corp.*, No. 13-cv-351 (E.D. Tex.).

STATEMENT OF JURISDICTION

The district court had jurisdiction pursuant to 28 U.S.C. §§ 1331, 1338(a). Following the verdict, the district court denied VirnetX's request for a permanent injunction, severed "VirnetX's claim for an ongoing royalty into a separate cause of action" (A75), and entered a "Final Judgment," noting that "all issues, between VirnetX and Apple, ***except future ongoing royalties***, if any, have been finally resolved" (A81).[1] VirnetX's request for an ongoing royalty remains pending. *VirnetX, Inc. v. Apple Inc.*, No. 13-cv-211 (E.D. Tex.).

Apple moved to amend the judgment, arguing that final judgment cannot enter until VirnetX's infringement claims—which include its ongoing royalty

[1] Emphases are added unless otherwise noted.

request—are finally resolved. A8163-8169; *see Warsaw Orthopedic, Inc. v. Nuvasive, Inc.*, 515 F. App'x 882, 882 (Fed. Cir. 2012) ("[T]he case is not 'final' because the district court has not yet determined ongoing royalties."). Apple also pointed out that severance of the ongoing royalty request was impermissible claim-splitting. A8166-8167. The court denied Apple's motion. A83-88.

Although Apple believes that the judgment was not truly final, Apple filed this appeal on July 1, 2013, to preserve its appellate rights. A8173-8174. If the judgment against Apple is deemed final, then this Court has jurisdiction under 28 U.S.C. § 1295(a)(1); otherwise, the Court lacks appellate jurisdiction.

STATEMENT OF ISSUES

1. Whether the infringement judgment for the '135 and '151 patents should be reversed or vacated, where there is no substantial evidence that Apple's VPN On Demand feature "determin[es] whether" a DNS request corresponds to a secure target or initiates a secure or encrypted channel "between" the client and target devices.

2. Whether the infringement judgment for the '504 and '211 patents should be reversed or vacated, where the district court erred in construing "domain name" and "secure communication link" and there is no substantial evidence that Apple's FaceTime servers provide "direct" communications.

3. Whether the judgment of no invalidity should be reversed, where a reasonable jury could only have found each asserted claim to be anticipated.

4. Whether a new trial should be granted because the district court erred in excluding evidence of reexamination proceedings showing Apple's good-faith belief of invalidity, which is relevant to inducement.

5. Whether the damages award should be reversed or remanded for a new trial.

STATEMENT OF THE CASE

VirnetX sued Apple for infringement in the Eastern District of Texas. A7172. VirnetX accused Apple's VPN On Demand feature in certain iPad, iPhone, and iPod touch devices of infringing the '135 and '151 patents and Apple's FaceTime servers of infringing the '504 and '211 patents. A7174, A7176. A jury found that Apple infringed all four patents and that no infringed claim was invalid, and awarded $368,160,000 in damages. A240-241.

Apple moved for judgment as a matter of law ("JMOL") or a new trial with respect to infringement, invalidity, and damages. A8046-8094. The court denied Apple's motion. *VirnetX Inc. v. Apple Inc.*, 925 F. Supp. 2d 816 (E.D. Tex. 2013) (A37-67). The court awarded VirnetX pre- and post-judgment interest, and post-verdict damages at the rate of $330,201 per day. A68-70. The court denied

VirnetX's requests for an injunction and attorney's fees (A70-79), neither of which VirnetX has appealed.

STATEMENT OF FACTS

A. The Parties

1. Apple

Since visionary inventor Steve Jobs co-founded the company in 1976, Apple has developed products that have revolutionized entire product categories. In the 1990s, Apple focused on providing high-quality Macintosh ("Mac") desktop computers and laptops. A1804:16-19.

In 2001, Apple introduced iPod, a portable media player that far exceeded the capabilities of other music storage devices. A1806:6-13. In 2007, Apple introduced iPhone, a smartphone with a touchscreen display that was capable of connecting to the Internet. A1806:14-20. And in 2010, Apple introduced iPad, a tablet computer that allows users to check email, browse the Internet, read books, watch movies, listen to music, and much more. A1807:4-25.

Apple's mobile devices run an Apple-designed operating system called "iOS." The features at issue here, VPN On Demand and FaceTime, are two among the hundreds of software features in Apple's Mac computers and iOS mobile devices.

2. VirnetX

VirnetX is a Delaware corporation based in Nevada. A7161; A7164-7165. VirnetX has been developing a single software product, “Gabriel,” which has never been sold. A1015:10-1016:8; A1263:2-23; A1907:7-12. Instead of marketing its software, VirnetX conducts an aggressive “pure IP licensing” campaign (A1113:1-3), seeking to license its patents to large companies, including Google, Cisco, Sony, Samsung, Verizon, and IBM (A1258:18-1260:10).

The applications that ultimately issued as the asserted patents were originally assigned to Science Applications International Corporation (“SAIC”). A1103:2-15. After failing to license its alleged inventions to other companies (A1104:3-24; A1136:12-1137:9), SAIC transferred the patents-in-suit to VirnetX (A1229:13-16).

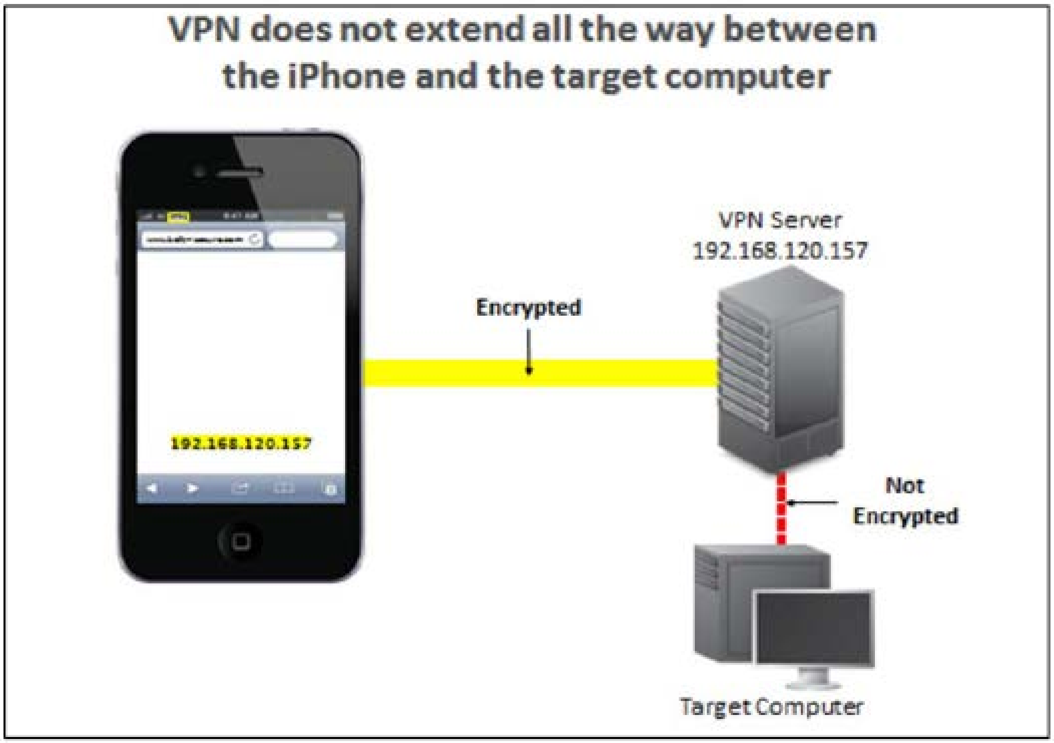
B. Background Regarding VPN On Demand

1. VPN On Demand’s “Always” mode

The iOS software installed on Apple’s accused mobile devices includes the “VPN On Demand” feature, which enables users to connect to user-selected websites or domains through a virtual private network (“VPN”). Although Apple’s Mac computers also include VPN On Demand, VirnetX did not accuse those products of infringement.

VPN On Demand operates when a user populates a configuration file with a list of "domain names" (*e.g.*, apple.com) to which the user would like to connect using a VPN. A1377:16-1378:22. When the user enters a website name into an Internet browser, for example, VPN On Demand compares the requested website name to the domain names in the configuration file. A1523:16-1524:18. In the accused "Always" mode,[2] if VPN On Demand determines that the requested website matches a domain name in the configuration file, it initiates an encrypted connection (shown in yellow) between the Apple mobile device and a specified VPN server, which is situated between the mobile device and the computer hosting the requested domain:

[2] VPN On Demand can operate in three modes: "Always," "If Needed," and "Never." VirnetX accused only the "Always" mode of infringement. A1566:14-22.



A8053; *see* A1388:12-24; A1421:9-18; A1509:19-1518:20. VPN On Demand does not, however, provide any encryption between the VPN server and the computer to which the user is attempting to connect (shown in red). A1379:9-12; A1392:4-18.

VPN On Demand will establish an encrypted connection ***only*** when the requested website or domain corresponds to a domain name in the configuration file, regardless of whether the requested website or domain itself is secure. Thus, even if the requested website or domain ***is*** secure, VPN On Demand will ***not*** initiate an encrypted channel unless the corresponding domain name is in the configuration file. A1520:1-22. Conversely, VPN On Demand ***will*** initiate an encrypted channel if the requested domain name is in the configuration file, even if the requested website or domain itself is ***not*** secure. A1514:5-1516:18.

2. VirnetX's asserted claims

VirnetX accused Apple's mobile devices of infringing claims 1, 3, 7, and 8 of the '135 patent and claims 1 and 13 of the '151 patent, which claim methods, devices, and software for enabling secure communications between computers.

The asserted claims all share a common limitation of "***determining whether*** [a] DNS request … is requesting access to a ***secure***" website or server.[3] A303(47:27-28); *see* A450(46:59-60) ("[a] data processing device … that … performs the steps of … ***determining whether*** the intercepted DNS request corresponds to ***a secure server***"). If the method or device determines that the DNS request is ***not*** seeking access to a secure target, then it passes the DNS request along as is normally done for Internet communications. A303(47:36-41); A450-451(46:61-64, 48:24-26). But if the method or device determines that the DNS request is requesting access to a ***secure*** website or server, the claimed invention "automatically initiat[es]" a VPN (in the '135 patent) or an "encrypted" or "secure" channel (in the '151 patent) between the client and the secure target. As named inventor Robert Short explained, the claimed invention is that "if the user

[3] A DNS (domain name service) request is a request that a particular lookup service return the Internet Protocol ("IP") address (*e.g.*, 162.18.254.1) corresponding to a given domain name (*e.g.*, www.apple.com). A31.

makes a request ***for a secure website***, the technology ***must do one thing***. And if the user makes a request ***for an unsecure website***, the technology ***must do another thing***." A1158:24-1159:4; *see* A1505:12-22.

Each asserted claim of the '135 and '151 patents also requires initiating a VPN or secure channel "***between***" the client and the secure target. A303(47:29-33) ("in response to determining that the DNS request … is requesting access to a secure target web site, automatically initiating the VPN ***between*** the client computer and the target computer"); A450(46:66-67) ("when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel ***between*** the client and the secure server"); A451(48:27-29). The district court construed "between [A] and [B]" to mean "extending from [A] to [B]," such that the claimed VPN or secure channel must extend all the way from the client to the secure target. A26.

C. Background Regarding FaceTime

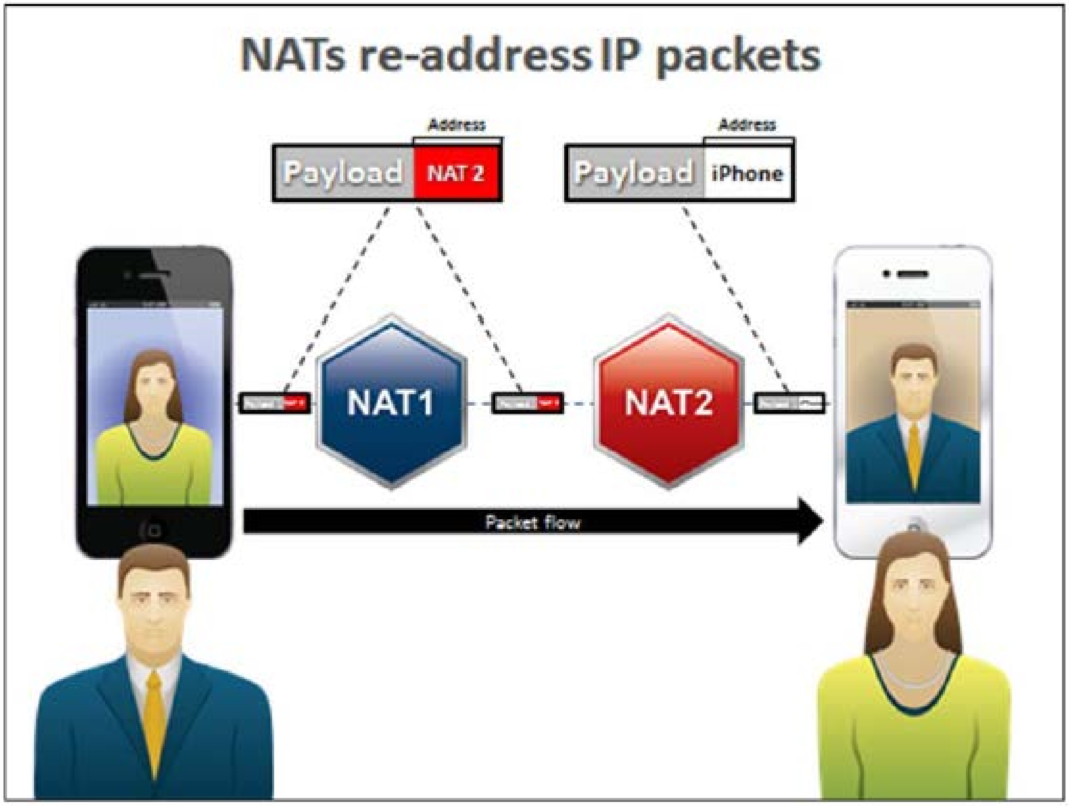
1. Apple's FaceTime feature

FaceTime is a video calling application developed by Apple engineers and first offered as a feature on iPhone 4 in 2010. A1809:16-18, A1812:13-15, A1813:22-24; A15216. A user initiates a FaceTime call by selecting the intended recipient's phone number or email address. A15022; A15224. An invitation "that includes the phone number [or email address] of the intended receiver of the

FaceTime video call" is then "sent to an Apple [FaceTime] server." A15216; *see* A1824:24-1825:4. The FaceTime server forwards the invitation to a network address translator ("NAT"), which readdresses and sends the invitation to the receiving device. A1825:2-4. The receiving device may accept the invitation to begin the FaceTime call. A15216; A1825:5-8.

Once a FaceTime call is underway, audio and video information is exchanged over the Internet without passing through the FaceTime servers. A1820:4-8; A1825:9-23. This information is broken down into small segments called "packets," which contain audio and video data (the "payload") as well as source and destination addresses so that the packets can be properly routed over the Internet. A1820:8-16.

Each data packet is addressed not to the private address of the receiving device, but to the public address of a NAT located between the sending and receiving devices. A1821:9-25. The NAT readdresses those packets to the receiving device's private address and sends them to that device:



A8057; *see* A1821:14-25; A15212.

Within the data packets transferred between FaceTime users, the audio and video content is encrypted. A15167; A1465:17-20. However, because the source and destination addresses of the data packets are not encrypted, eavesdroppers can determine the source and destination addresses of the packets as they travel between Apple devices. A1601:21-1602:7; A1467:16-1468:3.

2. VirnetX's asserted claims

VirnetX accused Apple's FaceTime servers (not its Mac computers or iOS mobile devices) of infringing claims 1, 2, 5, 16, 21, and 27 of the '504 patent and claims 36, 37, 47, and 51 of the ’211 patent. A1443:6-10; A1469:22-1470:3; A1481:6-10.

The asserted claims are directed to systems and software for providing “a domain name service for establishing a secure communication link.” A386(55:49-50); *see* A528(57:38-46). A “domain name service” is a lookup service that returns an IP address (*e.g.*, 162.18.254.1) for a requested domain name (*e.g.*, www.apple.com). A15. To facilitate this, each claim requires “stor[ing] a plurality of ***domain names*** and corresponding network addresses.” A386(55:52-53); A528(57:41-43).

Each asserted claim also requires the system or software to be capable of establishing a “***secure communication link***.” A386(55:54-56); A528(57:43-46). The common specification explains that the “secure communication link” must provide not only ***data security*** (concealing the data transmitted between users), but also ***anonymity*** (concealing the data’s source and destination). A359(1:52-54). To achieve both security features, the specification describes the claimed invention as “us[ing] a unique two-layer encryption format.” A360(3:13-16). “The message payload is hidden behind an inner layer of encryption” (A360(4:5-7)), while an “outer layer of encryption” conceals the communication’s “true destination” (A360(3:33-43)).

D. District Court Proceedings

1. Claim construction

VirnetX asked the district court to construe "domain name" in the '504 and '211 patents to mean "a name corresponding to an IP address." A5019. Apple argued, consistent with the term's plain meaning and use in the patents, that "domain name" means "a ***hierarchical*** sequence of words in decreasing order of specificity that corresponds to a numerical IP address." A6298-6299. The court adopted VirnetX's proposed construction, incorporating its reasoning from a separate case, *VirnetX, Inc. v. Microsoft Corp.*, 2009 WL 2370727 (E.D. Tex. July 30, 2009). A16.

VirnetX also urged the court to construe "secure communication link" in the '504 and '211 patents to mean "an encrypted communication link," while Apple argued that the term referred to a "virtual private network communication link"—a construction that would have required the communication link (i) to be ***direct*** and (ii) to provide both ***data security*** and ***anonymity***.[4] A5015. Adopting neither proposal, the court construed "secure communication link" as "a direct

[4] The court adopted Apple's proposed construction of "virtual private network" in the '135 patent as "a network of computers which privately and ***directly*** communicate with each other by encrypting traffic on insecure paths between the computers where the communication is both ***secure*** and ***anonymous***." A8.

communication link that provides data security through encryption." A7992; *see* A13.

2. Trial

Infringement. VirnetX presented its infringement case through its expert, Mark Jones. Regarding VPN On Demand, Jones opined that Apple's products "determin[e] whether" a DNS request seeks access to a secure website or server when VPN On Demand compares the requested domain name to the configuration file's list of domain names. A1395:6-10, A1510:1-24. Jones conceded, however, that VPN On Demand does not determine whether the requested domain itself is ***secure*** and that it initiates a VPN for ***any*** domain listed in the configuration file, even if it does not correspond to a secure website or server. A1510:21-24, A1521:20-1524:23. Jones also acknowledged that VPN On Demand encrypts communications only "between the iPhone and the VPN server" (A1392:4-13); he merely ***assumed*** that separate "private networks" would provide security or encryption between the VPN server and the target computer (A1379:9-14).

Regarding FaceTime, Jones testified that Apple's FaceTime servers "store a plurality of domain names" because they "store the phone numbers and e-mail addresses." A1460:9-15. He also opined that FaceTime supports establishing a "secure communication link," but only because "the audio and video streams are encrypted." A1455:21-1456:7. And while Jones agreed that "direct" communication requires "direct addressability" (A1543:3-5), he admitted that FaceTime communications travel through NATs, which readdress data packets by replacing the NAT's public address with the receiving device's private address (A1538:14-17, A1541:4-13).

***Invalidity*.** Apple demonstrated that all asserted claims were anticipated by a prior-art publication ("Kiuchi") describing a system that "provides secure HTTP communication mechanisms" over the Internet. A15008. Apple's invalidity expert, Peter Alexander, detailed how Kiuchi discloses each claim limitation. A2129:16-2130:3. VirnetX's expert Jones did not dispute that Kiuchi discloses most claim limitations; of the few he disputed, his primary assertion was that the communications in Kiuchi were not "direct." *E.g.*, A2333:10-2334:10, A2343:14-2344:4, A2347:20-2348:13.

***Induced Infringement*.** As part of its inducement case, VirnetX sought to establish that Apple knew of or was willfully blind to infringement by proffering deposition testimony of Apple engineers. A1735:18-1760:12. The district court precluded Apple from responding with evidence that Apple initiated reexamination proceedings—in which all asserted claims stand rejected—as evidence of Apple's good-faith belief that VirnetX's patents are invalid. A8002-8006.

***Damages*.** VirnetX's damages expert, Roy Weinstein, asserted three reasonable royalty theories. Apple moved to exclude all three under *Daubert v.*

Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579 (1993), but the district court overruled Apple's objections. A7661-7675; A8070-8091; A32-33; A1722:21-1723:8.

(1) Weinstein's first approach—which he avoided calling an "entire market value" approach—nonetheless included the entire market value of certain Apple products in the royalty base. Weinstein included the lowest price for ***each product model***, without attempting to apportion out unclaimed features. *E.g.*, A1614:24-1615:17, A1616:14-20, A1618:20-21, A1699:16-19. Weinstein did not apportion revenue due specifically to the accused features (FaceTime and VPN On Demand), much less to the specific aspects of those features that VirnetX accused of infringement. A1667:11-25. Weinstein's royalty base exceeded $70 billion. A1622:19-24, A1623:16-20.

Weinstein then applied a 1% royalty rate. He relied heavily on VirnetX's purported "policy" of licensing its patents for 1%-2% of the entire value of products sold (A1595:6-19), despite admitting that the policy represented simply what VirnetX "hoped for" (A1652:23-1653:13). Weinstein's rate yielded a $708 million demand (A1644:8-12), consisting of $566 million for devices including both FaceTime and VPN On Demand and $142 million for devices including only VPN On Demand (A1622:19-1623:6, A1623:14-1624:22).

(2) Weinstein offered a second damages method for FaceTime alone, which he purported to base on a mathematical model developed by economist John Nash, though he acknowledged that Nash never applied the model to patent licensing. A1629:9-22. Weinstein testified that, under his "Nash bargaining solution," "the parties split between themselves the incremental or additional profits that are associated with the use of the patented technology." A1630:4-15; *see* A1632:9-17. Weinstein believed—based on a calculation of revenue from a ***front-facing camera*** on Apple's mobile devices, a feature not directly linked to the claimed invention (A1710:4-8)—that the "profit associated with FaceTime" amounted to $1.3 billion (A7882:11-12).

Weinstein asserted that VirnetX would have received 45% of that profit (A1633:11-17), a figure he arrived at by arbitrarily taking "a 10 percent cut" from a 50/50 starting point (A1708:11-1709:4). Forty-five percent of $1.3 billion yielded a total FaceTime damages amount of $588 million under Weinstein's "Nash bargaining solution" theory. A1636:18-1638:14.

(3) Weinstein offered a third theory for FaceTime, which he called an "entire market value rule" approach. He initially claimed that FaceTime drove demand for Apple's iOS products (A1639:10-17), but later admitted that FaceTime and VPN On Demand only "contribute[d] to" Apple's sales. A1622:21-1663:3. Weinstein cited an Apple customer survey showing that, of the [redacted]

[redacted]

(A15292), 18% chose FaceTime as the "most desired" new feature. A1640:16-1641:5. Weinstein extrapolated from this that 18% of ***all*** iOS device sales would not have happened without FaceTime. A1641:8-21, A1711:13-1712:8. He ignored another survey showing that only 3% of iPhone customers listed FaceTime as the most desired feature. A1712:22-1713:11.

Weinstein did not attempt to determine either a royalty base or rate; instead, invoking the survey data, he concluded that 18% of Apple's mobile device revenue ($914 million) was attributable to FaceTime. A1641:8-1642:5. Then he purported to derive Apple's ***profit*** by deducting an unexplained amount for Apple's costs and applying the ***45%*** rate arbitrarily chosen for his "Nash bargaining solution" theory. A1642:6-11. Using this approach, Weinstein arrived at a figure of $5.13 per unit and a total of $606 million for FaceTime damages. A1641:22-1642:1, A1642:2-11, A1643:6-16.

Over Apple's objection (A2466:11-2468:21), the court instructed the jury that it could include "the value of the entire apparatus or product" in a royalty base if "the product in question ***constitutes the smallest saleable unit containing the patented feature***." A2515:24-A2516:9.

3. Post-verdict motions

The jury found all four patents infringed and not invalid, and awarded damages of $368,160,000. Apple moved for JMOL or a new trial, which the district court denied.

With respect to the ’135 and ’151 patents, the court rejected Apple’s non-infringement arguments, stating that the claims merely require a determination “whether a VPN is needed” (A40), even though the claims specify “***determining whether*** the DNS request … is requesting access to a ***secure web site***” (A303(47:27-28)). The court also rejected Apple’s argument that VPN On Demand does not provide a secure connection “***between***” the client and target devices, even though VPN On Demand encrypts communications only between the client (*e.g.*, iPhone) and a VPN server, not between the VPN server and the target device, as the claim construction requires. A43.

The court also rejected Apple’s arguments regarding the ’504 and ’211 patents. Although it was undisputed that Apple’s FaceTime communications travel through NATs, which change the IP addresses of the data packets (A1538:14-17, A1541:4-13), the court ruled that a “direct” communication encompassed this situation, because a NAT “merely translates addresses.” A46. And while Apple pointed out that VirnetX provided no evidence that a phone number or email address qualifies as a “domain name” (A8061), the court stated that email addresses and phone numbers are "domain names" because they have corresponding IP addresses. A47.

As for invalidity, the court ruled that there was sufficient evidence that Kiuchi did not satisfy at least one limitation of each asserted claim, including "direct" communication. A51-53.

Regarding damages, the court upheld its instruction that the jury could consider the entire value of Apple's iOS products. The court believed that there was "another alternative"—besides the entire market value rule—"when the jury could consider the entire value of a product." A64. The court described that "alternative" as "instances when the smallest saleable patent-practicing unit is the entire product" and "[d]epending on the claim language of a patent, it is foreseeable that an entire product is required to practice the invention." A64-65. In the court's view, the fact that a patent was practiced by "a single assembly or … a complete machine" justified basing damages on the machine's entire market value, even though the machine contained numerous unclaimed features. A65.

The court similarly ruled that VirnetX had not violated the entire market value rule because it provided "some evidence that the infringing features practicing the patented inventions necessarily utilized other aspects of the accused devices." A57-58. The court believed that such evidence "demonstrate[d] a close relation between the accused devices and the patented inventions." A57. The court also ruled that VirnetX's 1% royalty rate and "Nash bargaining solution" theory were adequately supported. A58-62.

SUMMARY OF ARGUMENT

1. The infringement judgment for the '135 and '151 patents should be reversed or vacated. Apple's VPN On Demand does not "***determin[e] whether***" a DNS request corresponds to a secure website or server (and is not even capable of doing so). Rather, it initiates a VPN whenever a requested domain name matches an entry in a user-created configuration file, regardless of whether the target website or server is secure. Nor does VPN On Demand initiate a secure or encrypted channel "***between***" the client and target devices; it undisputedly provides no security or encryption beyond the VPN server.

2. The infringement judgment for the '504 and '211 patents should also be reversed or vacated. When "domain name" is properly construed according to its plain meaning, it requires a ***hierarchical*** sequence of words (*e.g.*, www.apple.com). But Apple's FaceTime servers store only phone numbers and email addresses, not "domain names." The district court also erred in construing "secure communication link" as not requiring anonymity, contrary to the specification's clear instruction. Apple's FaceTime servers do not provide a "secure communication link" because FaceTime communications (i) are not

anonymous and (ii) travel through NATs, which readdress the data packets and thus preclude "direct" communication with the receiving device.

3. The judgment of no invalidity should be reversed for all four patents-in-suit. Apple demonstrated that Kiuchi discloses every claim limitation—most of which were not even disputed by VirnetX.

4. Alternatively, a new trial should be granted on all infringement claims. Apple sought to introduce evidence that, after learning of VirnetX's patents, it initiated reexamination proceedings in which all asserted claims stand rejected. That evidence would have shown Apple's good-faith belief of invalidity and rebutted any claim of intent to induce infringement. The court's exclusion of that evidence was erroneous and highly prejudicial.

5. Finally, the damages award should be reversed or vacated. The court erroneously instructed the jury that it could use the entire value of Apple's products as a royalty base even if VirnetX did not satisfy the entire market value rule. VirnetX's asserted royalty base unquestionably included the entire market value of Apple products and VirnetX did not even purport to satisfy the entire market value rule. VirnetX also relied on licenses not commensurate with the accused infringement and on a speculative and arbitrary "Nash bargaining solution" theory. Part or all of VirnetX's damages testimony should have been stricken, warranting either JMOL or a new trial.

STANDARD OF REVIEW

This Court reviews JMOL, new trial, and evidentiary rulings under regional circuit law. *Verizon Servs. Corp. v. Cox Fibernet Va., Inc.*, 602 F.3d 1325, 1331 (Fed. Cir. 2010). The Fifth Circuit reviews the denial of JMOL *de novo*, making JMOL appropriate if "the Court believes that reasonable men could not arrive at a contrary verdict." *Medical Care Am., Inc. v. National Union Fire Ins. Co. of Pittsburgh, Pa.*, 341 F.3d 415, 420 (5th Cir. 2003). This Court has interpreted that standard to mean that the jury's determination must be supported by substantial evidence. *ACCO Brands, Inc. v. ABA Locks Mfrs. Co.*, 501 F.3d 1307, 1312 (Fed. Cir. 2007).

This Court reviews claim construction *de novo*. *Cybor Corp. v. FAS Techs., Inc.*, 138 F.3d 1448, 1454-1455 (Fed. Cir. 1998) (en banc). Infringement and anticipation are questions of fact reviewed for substantial evidence. *B. Braun Med., Inc. v. Abbott Labs.*, 124 F.3d 1419, 1423 (Fed. Cir. 1997); *Finisar Corp. v. DirecTV Group, Inc.*, 523 F.3d 1323, 1334 (Fed. Cir. 2008). The patentee has the burden of proving damages, which are reviewed for substantial evidence. *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1310, 1324 (Fed. Cir. 2009).

Evidentiary decisions are normally reviewed for an abuse of discretion. *Sulzer Textil A.G. v. Picanol N.V.*, 358 F.3d 1356, 1363 (Fed. Cir. 2004). But "where a district court rules, as a matter of patent law, that a party is precluded from introducing evidence," this Court applies *de novo* review. *Id.* A legal error is an abuse of discretion. *Love v. Tyson Foods, Inc.*, 677 F.3d 258, 262 (5th Cir. 2012).

ARGUMENT

I. THE JUDGMENT THAT APPLE'S VPN ON DEMAND PRODUCTS INFRINGE THE '135 AND '151 PATENTS SHOULD BE REVERSED.

A. There Is No Substantial Evidence That VPN On Demand "Determin[es] Whether" A DNS Request Corresponds To A Secure Target.

Each asserted claim of the '135 and '151 patents requires "***determining whether***" a DNS request corresponds to a secure target ("secure web site" or "secure server"). A303(47:27-28); A450(46:59-60); A451(48:22-23). If that determination indicates that the DNS request is for access to a secure target, then the claimed invention automatically creates a secure or encrypted link for communicating with the secure target. A303(47:29-32); A450(46:65-67); A451(48:27-29).[5]

VirnetX contended that the "determining whether" limitations were met literally (A1370:13-16, A1395:6-10), but provided no evidence that Apple's VPN On Demand actually determines whether a DNS request is seeking access to a secure website. Instead, the undisputed evidence demonstrated that VPN On

[5] The district court construed "secure web site" as "a web site that requires authorization for access and that can communicate in a VPN." A21; *see* A23-24 (similar construction for "secure server").

Demand ***cannot*** make such a determination. VPN On Demand initiates a VPN connection whenever the requested domain name is listed in a user-created "configuration file." A1506:7-1508:3(Jones); A1872:10-21, A1878:5-1880:18, A1891:8-1900:25(Kelly). VirnetX's expert Jones conceded that VPN On Demand will initiate a VPN connection for ***any*** domain name in the configuration file, "whether it's a secure website or non-secure website," and that VPN On Demand will ***not*** initiate a VPN for a secure domain name that is not listed in the configuration file. A1523:2-12; *see* A1388:7-24; A1510:21-24, A1520:12-22, A1521:20-1524:23. Thus, VPN On Demand never determines whether a requested domain name corresponds to a secure website or not.

The district court nevertheless denied JMOL, stating: "There is no requirement in the claims … that the security of the requested website or server be verified. Instead, the claims merely require that the feature ascertain whether a VPN is needed." A40. But that reasoning cannot be reconciled with the claim language, which expressly requires "***determining whether*** the DNS request … is requesting access to ***a secure server***" (A303(47:27-28)) or "***determining whether*** the intercepted DNS request ***corresponds to a secure server***" (A302-303(46:59-60, 48:22-23)). Neither party asked the district court to construe the "determining whether" language (A38), and the jury was instructed to apply its ordinary meaning (A2495:21-24). The jury's verdict must accordingly be judged against that ordinary meaning, not a new (and erroneous) interpretation articulated ***after*** the verdict. *See Hewlett-Packard Co. v. Mustek Sys., Inc.*, 340 F.3d 1314, 1320-1321 (Fed. Cir. 2003) ("[W]here the parties and the district court elect to provide the jury only with the claim language … it is too late at the JMOL stage to argue for or adopt a new and more detailed interpretation of the claim language and test the jury verdict by that new and more detailed interpretation."). Even under the district court's post-verdict interpretation, VPN On Demand does not "ascertain whether a VPN is needed"; it initiates a VPN ***if and only if*** the requested domain name is listed in the configuration file—regardless of whether a VPN is "needed."

The district court further stated that "Apple's VPN On Demand feature can be configured to infringe." A40. That is also incorrect. Regardless of how a user sets up the configuration file, VPN On Demand ***is incapable of*** "determining whether" a requested domain name corresponds to a secure website. No evidence suggested otherwise. VPN On Demand will ***only*** ever initiate a VPN when the requested domain name is in the configuration file, regardless of the target website's security. *See supra* pp. 6-9, 25-26.

VirnetX relied on certain marketing materials that mention access to a "private" or "corporate" network. A1395:11-23(Jones); A15235-15249; A15025-15118; A15489-15492; A15250-15256. While those documents provide examples of networks with which users may choose to initiate VPN connections, they do not change the fact that VPN On Demand cannot "***determine whether***" the target website or server is secure. Documents cannot change how VPN On Demand actually operates. *See Phillips Petroleum Co. v. Huntsman Polymers Corp.*, 157 F.3d 866, 877 (Fed. Cir. 1998) (no infringement despite statements using claim language in "internal documents and product literature"); *CMI Corp. v. Metro. Enters., Inc.*, 534 F.2d 874, 883 (10th Cir. 1976) ("[S]tatements in an alleged infringer's advertising … cannot serve to controvert what is clearly demonstrated to be the actual fact."); *see also Scantibodies Lab., Inc. v. Immutopics, Inc.*, 374 F. App'x 968, 971 (Fed. Cir. 2010) ("The use of language in marketing materials often means something quite different from the language used in a patent.").

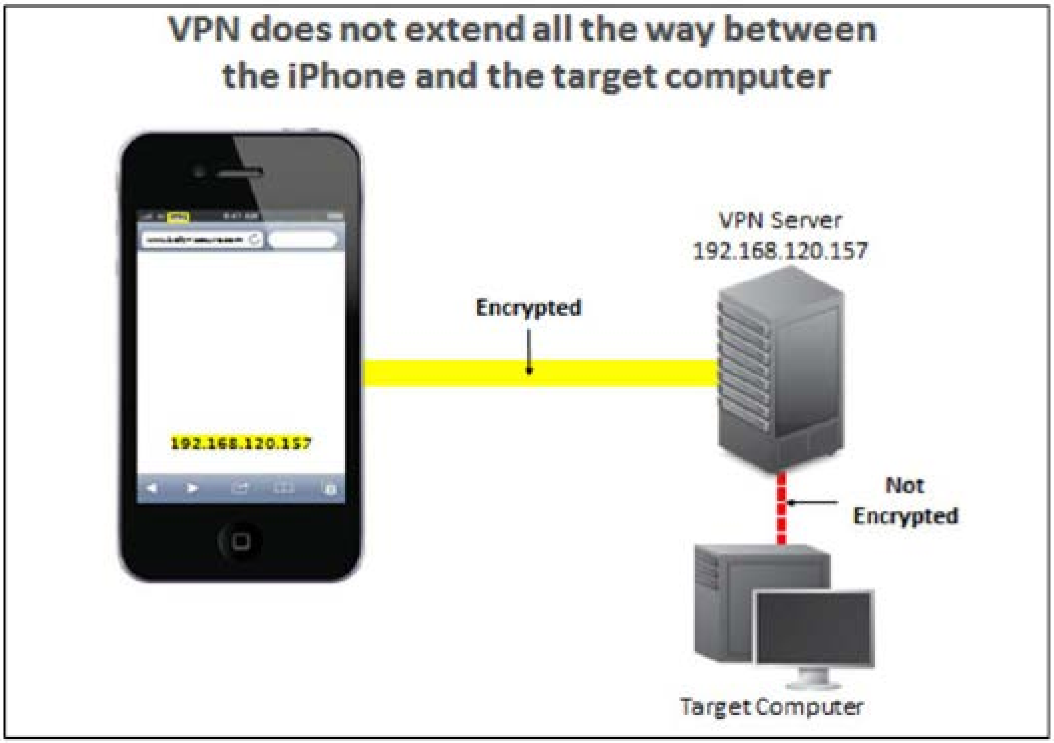
B. There Is No Substantial Evidence That VPN On Demand Initiates A VPN, Or A Secure Or Encrypted Channel, "*Between*" The Client And Target.

1. VPN On Demand does not create a VPN or secure channel "*between*" the client and target.

Claim 13 of the '151 patent requires "creating a secure channel ***between*** the client and the secure server." A451(48:28-29). Claim 1 of the '135 patent similarly recites "creating a virtual private network (VPN) ***between*** a client computer and a target computer." A303(47:20-21); *see* A303(47:31-32). The district court construed "VPN" to mean "a network of computers which privately and directly communicate with each other by encrypting traffic on insecure communication paths ***between*** the computers where the communication is ***both***

secure and anonymous" (A8), and construed "between" to mean "extend[ing] from" the client to the target (A26). To prove infringement, VirnetX was therefore required to show that VPN On Demand establishes a communication path that is "secure" (and also "anonymous" for the '135 patent) all the way from the client to the target.

The undisputed evidence demonstrated, however, that VPN On Demand does ***not*** provide security or anonymity "between" the client and target. Jones testified ***only*** that VPN On Demand encrypts communications "between the iPhone [the client] and the VPN server" (shown in yellow below) (A1392:4-18); but VPN On Demand undisputedly does not initiate any connection, or encrypt any data or addresses, between the VPN server and the target (shown in red below):



A8053.[6] Accordingly, VPN On Demand does not provide security or anonymity "between" the client and target.

In denying JMOL, the district court summarily stated that Jones "detailed how a secure and anonymous communication between computers is established by the feature." A43 (citing A1380:17-1383:12, A1400:18-1401:23). That is simply wrong. Jones testified as to encryption "between the iPhone and the VPN server" only. A1392:4-18; *see* A1378:23-1379:2, A1379:3-8, A1401:1-11, A1428:18-23. For the path from the VPN server to the target device, Jones's infringement

[6] Apple's expert testified without contradiction that, when using VPN On Demand to connect over the Internet to a website, "[a hacker] … can see the traffic on the – over on the right side of the VPN server" (A1997:19-24) and VPN On Demand "ha[s] not made secure access" (A1998:1-14).

opinion relied entirely on the ***assumption*** that private networks—not Apple—could provide security and anonymity. *See, e.g.*, A1379:9-14 ("Q. Okay. Now, once the connection gets inside the confines of the company office, is the data necessarily encrypted in that portion? A. ***Not necessarily***. If that's a private network of the company … the company would have configured that to be secure."); A1400:18-1401:23 (hypothesizing that "the company has configured that network and … made a secure network there"); A1428:2-23 (testifying that the "between" limitation was literally satisfied because "between the last half … it's secured because that's a private network of security"). Assumptions not based in evidence, however, are legally insufficient to prove infringement. *See Meyer Intellectual Props. Ltd. v. Bodum, Inc*., 690 F.3d 1354, 1370 (Fed. Cir. 2012) ("We find it troubling that the district court based its direct infringement analysis on what it assumed happened rather than on actual evidence in the record. This assumption contradicts our well-established law that a patentee must prove infringement by a preponderance of the evidence.").

Additionally, Jones's assumptions fail on their own terms. While he assumed that VPN On Demand only interacts with private networks, VPN On Demand will initiate a VPN for ***any*** domain name listed in the configuration file—whether public, private, unsecure, or secure. A1523:2-1524:18(Jones); A1881:8-1884:13(Kelly). And even as to private networks, both Jones and named inventor

Short conceded that they are not necessarily secure or anonymous. A1379:9-14(Jones); A1160:24-1161:8(Short) (agreeing that private networks may not be secure, depending on "how [they're] set up and protected"); A1164:12-22(Short) (agreeing that "there could be interlopers on private networks" such as "hostile employees sniffing around on the network"). Thus, while VirnetX alleged that ***private network owners*** could provide for secure and anonymous communications, there was simply no evidence—let alone substantial evidence—that ***Apple's VPN On Demand*** initiates a secure channel or VPN that extends to the target computer.[7]

2. VirnetX's theory of equivalents regarding claim 1 of the '151 patent is legally insufficient.

Claim 1 of the '151 patent requires "initiating an encrypted channel" (rather than a secure channel) "***between*** the client and the secure server." A450(46:66-67). Jones conceded that VPN On Demand does not literally practice this element because the "private network" between the VPN server and the target server is "not necessarily encrypted." A1420:23-1421:18; *see* A31. Jones instead contended that this limitation was satisfied under the doctrine of equivalents (A1421:19-24), but his testimony was legally insufficient to sustain such a finding.

To begin with, VirnetX's "theory of equivalence would entirely vitiate a particular claim element." *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.,* 520

[7] Jones's assumption also contravenes the court's claim construction, which rejected VirnetX's suggestion that communications between the VPN server and target computer are "inherently secure." A26.

U.S. 17, 39 n.8 (1997). Jones asserted that VPN On Demand, which provides ***no encryption*** between the VPN server and the target server, is equivalent to a channel that ***is encrypted*** all the way to the target server. A1420:23-1424:25. A theory that equates a claim limitation with its opposite is a classic case of vitiation. *See Moore U.S.A., Inc. v. Standard Register Co.*, 229 F.3d 1091, 1106 (Fed. Cir. 2000).

VirnetX also failed to provide the "particularized testimony and linking argument" necessary to prove equivalence. *Amgen Inc. v. F. Hoffman-La Roche Ltd.*, 580 F.3d 1340, 1382 (Fed. Cir. 2009). Jones's "function, way, result" analysis was perfunctory; he stated that VPN On Demand "provid[es] data security through encryption on network paths that are vulnerable to eavesdroppers" and then concluded that "[t]he entire communication path between the client and the secure server is protected from potential eavesdroppers." A1424:4-25. But Jones failed to explain ***how*** VPN On Demand could encrypt or protect "[t]he entire communication path" when it provides no protection beyond the VPN server. Jones's assumption that private networks could protect the communication path beyond the VPN server fails to demonstrate equivalence, especially given his and inventor Short's concession that private networks are not necessarily secure. *See supra* pp. 31-32. Jones's conclusory opinion cannot support a finding of infringement by equivalents. *See Creative Internet Advertising Corp. v. Yahoo!, Inc.*, 476 F. App'x 724, 732-733 (Fed. Cir. 2011).

II. THE JUDGMENT THAT APPLE'S FACETIME SERVERS INFRINGE THE '504 AND '211 PATENTS SHOULD BE REVERSED.

A. Apple's FaceTime Servers Do Not Store "Domain Names."

Each asserted claim of the '504 and '211 patents requires storing "a plurality of ***domain names*** and corresponding network addresses." A386(55:52-53); A528(57:41-43). The district court ruled that a "domain name" could be ***any*** "name corresponding to an IP address." A16. That construction is inconsistent not only with testimony of VirnetX's own expert—who admitted that a device, such as a phone, "may have an IP address on the Internet but ***not*** have a domain name" (A7093(¶ 25))—but also with the plain meaning of the term. It was well-known at the time of the alleged inventions that "domain name" refers to "a ***hierarchical*** sequence of words in decreasing order of specificity that corresponds to a numerical IP address." A6298; *see* A6140 (VirnetX quoting a 1997 technical dictionary defining "domain name" as "[a]n address of a network connection that identifies the owner of that address in a ***hierarchical*** format: ***server.organization.type***"). Generally, a domain name like "www.apple.com" includes: (i) a top-level domain identifying the type of organization using the domain name (.com); (ii) a second-level domain identifying the organization itself (apple); and (iii) a host name identifying a particular computer or server within the network (“www” for a web server). *See Akamai Techs., Inc. v. Limelight Networks, Inc.*, 629 F.3d 1311, 1314 (Fed. Cir. 2010) (web addresses include “a domain name … (e.g., ‘www.cafc.uscourts.gov’)”), *vacated on other grounds*, 419 F. App’x 989 (Fed. Cir. 2011).

VirnetX conceded that the “typical domain name” contains such a “hierarchical syntax” (A7138; *see* A5019) and offered no evidence that “domain name” had any different meaning to skilled artisans. Instead, VirnetX merely argued—based on its expert’s conclusory opinion—that using a hierarchical sequence of words to identify an IP address is unnecessary because “a DNS proxy server does not need to provide answers for every domain name on the Internet.” A5020. But whether the claimed inventions ***could*** have stored something other than domain names is irrelevant; what the patents ***actually*** disclose and claim is storing “domain names,” not other information.

Consistent with the term’s ordinary meaning, the common specification discloses only “domain names” that include hierarchical sequences of words. *E.g.*, A378(39:9) (“Yahoo.com”); A378(39:29) (“Target.com”); A384(52:43-44) (“website.scom”; “website.com”). The specification also repeatedly refers to the hierarchical structure of “domain names” in describing the claimed inventions. *E.g.*, A383-384(49:25-29, 50:25-36, 52:35-44) (“top-level domain names”). Moreover, VirnetX conceded that ***every*** disclosed domain name has this hierarchical structure (A7138) and that the specification contains no special definition or clear disavowal that would justify departing from the ordinary meaning (A5020). *See Toshiba Corp. v. Imation Corp.*, 681 F.3d 1358, 1369 (Fed. Cir. 2012) ("Absent disclaimer or lexicography, the plain meaning of the claim controls.").

The district court nonetheless rejected Apple's proposed plain-meaning construction "for the same reasons stated in *Microsoft*." A16. In *Microsoft*, the district court refused to construe "domain name" to require a hierarchical structure because "[t]he specification's disclosure or omission of examples does not create limitations on claims." 2009 WL 2370727, at *8.[8] But Apple's proposed construction would not limit the ordinary meaning of the claim term based on examples in the specification. Rather, the specification "repeatedly and uniformly" applies the plain meaning of "domain name," even referring specifically to its hierarchical structure (*e.g.*, "top-level domain"). Nothing in the specification or elsewhere provides any basis for broadening "domain name" beyond its ordinary meaning. *See ICU Med., Inc. v. Alaris Med. Sys., Inc.*, 558 F.3d 1368, 1374 (Fed.

[8] In *Microsoft*, the district court also refused to construe "domain name" to require a hierarchical structure because "Microsoft relies largely on extrinsic evidence—including expert testimony and Microsoft's own technology tutorial—to support its contentions, which does not carry great weight in light of the fact that claim language provides guidance on the meaning of 'domain name.'" *Microsoft*, 2009 WL 2370727, at *8. It was inappropriate, however, to penalize ***Apple*** for arguments that ***Microsoft*** made in a separate litigation.

Cir. 2009) (construing "spike" to require a pointed tip, where the specification "repeatedly and uniformly" described only spikes with pointed tips and there was no evidence that the ordinary meaning of "spike" would include a non-pointed structure).

When "domain name" is properly construed, Apple's FaceTime servers do not infringe. Rather than storing "domain names" such as "www.apple.com," Apple's FaceTime servers store users' telephone numbers and email addresses, neither of which are hierarchical sequences of words in decreasing order of specificity. A1451:22-1452:12, A1460:9-15(Jones); A1953:7-1954:1(Kelly); A15216-15217. The Court should reverse the infringement judgment for the '504 and '211 patents, *see CVI/Beta Ventures, Inc. v. Tura LP*, 112 F.3d 1146, 1162 (Fed. Cir. 1997), or alternatively remand for a new trial under the proper construction, *see August Tech. Corp. v. Camtek, Ltd.*, 655 F.3d 1278, 1286 (Fed. Cir. 2011).

B. Apple's FaceTime Servers Cannot Establish The Claimed "Secure Communication Link."

While the district court correctly concluded that the claimed "secure communication link" must be ***direct***, it failed to recognize that the link must also provide ***anonymity***. *See* A6290-6292, A6297. Apple's FaceTime servers cannot establish such a "secure communication link" because they do not provide anonymity or allow for direct communication.

1. The court erred in construing "secure communication link" as not requiring anonymity.

Each asserted claim of the '504 and '211 patents requires that the system be capable of establishing a "secure communication link." A386(55:49-50, 55:54-56); A528(57:43-46). Apple proposed construing this term to mean "a virtual private network communication link," which would have required both data security and anonymity (A6290-6292, A6297), consistent with the court's construction of "virtual private network" in the '135 patent as requiring "communication [that] is ***both secure and anonymous***" (A8). Instead, however, the court construed "secure communication link" to mean "a direct communication link that provides data security through data encryption," but not anonymity. A13; A7638. That was error.

"The claims of the patent must be read in light of the specification's consistent emphasis on [the] fundamental feature of the invention." *Praxair, Inc. v. ATMI, Inc.*, 543 F.3d 1306, 1324 (Fed. Cir. 2008). Here, every portion of the specification indicates that the "secure communication link" established by the claimed invention must provide not only ***data security*** (concealing the data transmitted between users), but also ***anonymity*** (concealing the data's source and destination). A359(1:52-54).

The Background of the Invention notes that "[a] tremendous variety of methods have been proposed and implemented to provide security ***and anonymity*** for communications over the Internet." A359(1:32-3:10). It explains that secure communications must be "immune to eavesdropping" in two respects: the content of the transmitted information must be secret and the identities of the communicating parties must be protected. A359(1:40-47). As the patents explain, these two security requirements "may be called ***data security*** and ***anonymity***, respectively." A359(1:52-54).

The Summary of the Invention describes how the claimed invention achieves improved security by using "two-layer encryption." A359(2:66-3:2). First, the "inner layer of encryption" provides ***data security*** by protecting the "payload" (the data itself). A360(4:5-10). Second, the "outer layer of encryption" provides ***anonymity*** by concealing each data packet's "true destination." A360(3:13-4:13). The Summary of the Invention thus makes clear that the claimed invention establishes a secure communication link that provides both data security and anonymity. *See C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 864 (Fed. Cir. 2004) (noting that "statements that describe the invention as a whole," such as in the Summary of the Invention, "are more likely to support a limiting definition of a claim term").

Likewise, the Detailed Description uniformly discloses communication links providing anonymity as well as data security. For example, it explains that "the message payload is embedded behind an inner layer of encryption" and "[e]ach …

packet's true destination is concealed behind an outer layer of encryption." A360(3:33-34); A320(Fig. 2). ***Every*** disclosed embodiment includes a communication link that provides data security and anonymity. *E.g.*, A364-366(11:9-11, 11:34-36, 12:29-31, 12:39-44, 14:59-61, 15:55-58, 16:3-4, 16:18-20).

U.S. Patent No. 8,051,181 ("the '181 patent"), which is related to the '504 and '211 patents (A11), confirms that a "secure communication link" requires anonymity. The '181 patent clearly equates a "secure communication link" with a "virtual private network" link, which requires both data security and ***anonymity*** (A8). *See* A6747(abstract), A6800(6:56-58) ("The secure communication link is a virtual private network communication link over the computer network."); A6823(51:64-52:11) ("[T]he user can optionally specify that all communication links established over computer network 3302 are ***secure communication links***. Thus, anytime that a communication link is established, the link is a ***VPN link***."). Moreover, during prosecution of the '181 patent, VirnetX distinguished the claimed "secure communication link" from prior art because a skilled artisan "would not have considered the client and target [in the prior art] to be ***virtually on the same private network***." A7157. VirnetX thus used "secure communication link" synonymously with VPN link, which requires anonymity. *See Dayco Prods., Inc. v. Total Containment, Inc.*, 258 F.3d 1317, 1326 (Fed. Cir. 2001) (where related patents contain "similar … structure and wording," there is "no reason to construe the claims of [one patent] more narrowly than those of the [other]"); *Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1349 (Fed. Cir. 2004) ("[T]he prosecution history of one patent is relevant to an understanding of the scope of a common term in a second patent stemming from the same parent application.").

The district court accordingly erred by ignoring the specification's clear teaching, confirmed by the '181 patent, that the claimed "secure communication link" must provide ***both*** data security and anonymity. *See, e.g.*, *Eon-Net LP v. Flagstar Bancorp*, 653 F.3d 1314, 1321-1323 (Fed. Cir. 2011) ("the specification unequivocally compels" a limitation where it "repeatedly and consistently" described the claimed system in accordance with that limitation); *Poly-Am., L.P. v. GSE Lining Tech., Inc.*, 383 F.3d 1303, 1310 (Fed. Cir. 2004) (construing claims to include a limitation where the specification was "replete with references" to the limitation).

2. Under the correct construction, Apple's FaceTime servers do not establish a "secure communication link" because they do not provide anonymity.

FaceTime communications are protected by only an inner layer of encryption that provides data security for audio/video data, but not anonymity. Jones testified only that FaceTime's data content ("audio and video streams") is encrypted, while other information used to establish a FaceTime call is not.

A1465:17-20 (referring to "encrypted packets"); A1601:21-1602:7 ("Q. Okay. Now, might the devices … send some information to each other that is not encrypted? A. Yes. For example, the beginning of the communication in what's called a SIP, some SIP messages, when they're trying to establish a connection between the two phones, that's not encrypted."). Jones further explained that eavesdroppers could identify the particular Apple device to which a FaceTime call is directed—in other words, the communication is not anonymous. A1467:16-1468:3 ("[W]hen the phone wants to talk to the other one directly, it uses a transport address that will pass—that ***will allow it to identify that particular phone***. And that's what's sent over the Internet.").

Because Apple's FaceTime servers do not provide anonymity, they cannot establish the claimed "secure communication link," as properly construed. The infringement judgment for the '504 and '211 patents should accordingly be reversed, or at least remanded for a new trial under the correct construction.

3. Even under the district court's construction, Apple's FaceTime servers do not establish a "secure communication link" because they do not provide "direct" communication.

The district court correctly construed "secure communication link" to require a "***direct*** communication link." A13; A7992. The parties and experts agreed that a "direct" communication link does not require "a direct electromechanical connection," but instead requires (at a minimum) that the data be ***directly addressed*** to the intended recipient. A8 n.2; *see* A1542:25-1543:5(Jones); A1919:12-23(Kelly); A2651:7-20(VirnetX's counsel) ("[The] communication in the addressing context is not direct because the client computer addresses the message to the server. The server then readdresses the message to the target.").

The undisputed evidence demonstrated, however, that Apple's FaceTime servers do not establish such "direct" communication between user devices. As the district court noted, VirnetX "conceded" that FaceTime calls are not "direct" when they are sent through "relay server[s]." A46; *see* A1543:24-1544:8. And as VirnetX's expert Jones explained, FaceTime calls transmitted through relay servers are not "direct" because communications from one Apple device are not addressed to another Apple device, but to the relay server, which then readdresses those communications and passes them on to the receiving device. A1445:7-1446:1; A1565:5-12; *see* A1984:10-1985:8(Kelly).

Even when relay servers are not used, however, Apple's FaceTime servers still do not provide a "direct" communication link, because each Apple device's communications are not directly addressed to the other device, but to a NAT. There was no factual dispute on this point: ***Jones conceded*** that NATs readdress the incoming packets by replacing the NAT's public IP address with the target device's private IP address. A1538:14-17 ("Q. … So that packet coming out [of the NAT] is going to have a different IP address on it and destination IP address. Right? A. Yes."); A1541:4-13 (agreeing that the "destination IP address [of the transmitted packets] would change from one side of the NAT to the other"). Thus, because data packets from the sending device are addressed to either a relay server or a NAT—and undisputedly are ***not*** directly addressed to the target device—Apple's FaceTime servers do not provide a "direct" communication link.

The district court nevertheless ruled that NATs "do not impede direct communication" because they "merely translate[] addresses" and "allow[] for 'end-to-end communication between two devices.'" A46 (quoting A1462:1-1463:5). That ruling, however, conflicts with the court's claim construction and VirnetX's statements to the Patent Office on which the claim construction rested. VirnetX expressly disclaimed systems, like FaceTime, in which communications from a sending device are not directly addressed to the receiving device, but instead require an intermediate device that readdresses the communication to the receiving device's IP address. During prosecution, VirnetX distinguished a prior art reference ("Aventail") by claiming that Aventail "[did] not disclose a secure communication link, where data can be ***addressed to a target***, regardless of the location of the target." A7155. Instead, VirnetX contended, the client computer in Aventail first "transmits data to a SOCKS server … [which] then relays that data to a target computer on a private network on which the SOCKS server also resides."

A7157. Similar to VirnetX's description of Aventail, FaceTime communications are addressed to a NAT, rather than to an Apple device on the receiving end of the call, and thus do not "directly" address the target device. The district court was therefore wrong in stating that communications addressed to intermediate devices that "translate[] addresses" can be "direct" (A46), where the parties and the experts all agreed that the governing claim construction required the data (at a minimum) to be ***directly addressed*** to the intended recipient. A8 n.2; *see supra* pp. 42-43.

The district court also erred in relying on internal Apple documents as "evidence that a NAT operate[s] like a router, firewall or other similar servers and d[oes] not impede direct communication." A46. While some of Apple's documents describe communications through NATs as "direct peer-to-peer connections," that description merely refers to "direct … ***connections***" that are not made through relay servers; it says nothing about the "direct ***communication*** link" required by the claim construction. A13; *see* A15394; A15449; A15479; A15485; A1827:18-22(Gates); A1943:19-1944:1(Kelly). Nothing in Apple's documents indicates that FaceTime communications through NATs are ***communications directly addressed*** to receiving devices as the claim construction requires. In any event, word choice in Apple's documents cannot overcome the way that FaceTime actually and undisputedly works: FaceTime communications are addressed not directly to the receiving device, but to NATs that then readdress data packets to the receiving device. *See CMI*, 534 F.2d at 883; *see also Scantibodies*, 374 F. App'x at 971. That is decidedly not a "direct" communication and therefore, applying the district court's construction, not the "secure communication link" that the '504 and '211 patents require.

III. THE JUDGMENT OF NO INVALIDITY SHOULD BE REVERSED.

A. Kiuchi Anticipates The Asserted '135 Patent Claims.

For claims 1, 3, and 7 of the '135 patent, VirnetX contended that Kiuchi failed to disclose only the "virtual private network" limitation, which the district court construed as requiring (in relevant part) computers that "directly communicate" with each other (A6-8). *See* A2343:14-2344:12; A8125. No reasonable jury could have accepted that argument. Kiuchi teaches a VPN including a client computer that communicates directly with an origin server. A15010; A2119:17-2120:2(Alexander). The HTTP proxy computers located between the client and origin merely "forward[] requests to the origin server" (A15010); they do not readdress data packets and do not preclude direct communication, particularly given the district court's construction that "routers, firewalls, and similar servers that participate in typical network communication do not impede 'direct' communication" (A8 n.2; *see* A2119:12-2121:15(Alexander)).

For claim 8, VirnetX additionally contended that Kiuchi does not disclose "a DNS proxy server that passes through the [DNS] request." A2344:18-2345:9;

A8125. Although VirnetX's expert Jones testified that Kiuchi's client-side proxy "return[s] an error … but … doesn't pass through the [DNS] request" if the request corresponds to an unsecure website (A2345:12-2346:1), that conclusion is contradicted by the reference itself. Kiuchi clearly states that, if a DNS request does not correspond to a secure server, the proxy server "sends a status code which indicates an error. If a client-side proxy receives an error status, ***then it performs DNS lookup***, behaving like an ordinary HTTP/1.0 proxy." A15009; *see* A2109:23-25, A2128:8-19(Alexander). In other words, the error message causes the DNS request to be passed through the server—just as claim 8 requires.

B. Kiuchi Anticipates The Asserted '151 Patent Claims.

For the '151 patent, VirnetX argued that Kiuchi failed to disclose only two limitations: (i) a DNS request "sent by a client," and (ii) an "encrypted channel" or "secure channel" between the client and the secure server. A2389:1-7; A8125. Again, no reasonable jury could have agreed.

Kiuchi discloses DNS requests sent from a client-side proxy, which are DNS requests "sent by a client." A15009. Jones's attempt to distinguish a client-side proxy from "a client" (A2341:4-6) ignores that Kiuchi's client-side proxy performs precisely the same steps required of the claimed client computer (A2103:17-2129:4(Alexander)). Moreover, Apple's expert testified, without contradiction, that a client-side proxy can be a client computer even if the end-user is also a client computer. A2183:15-25, A2187:18-2188:6(Alexander); *cf.* A1418:22-24(Jones agreeing that the claimed "secure server" may also include proxy servers).

Kiuchi also discloses an "encrypted channel" (which is a type of "secure channel") between the client and the secure server. While Jones testified that Kiuchi does not disclose a single encrypted channel from the client-side proxy to the origin server (A2341:23-2342:22), that cannot be squared with Kiuchi's disclosure of an "end-to-end" encrypted channel along an entire path of communication (A15013 (disclosing communications "which assure end-to-end or individual security"); *see* A2112:3-8(Alexander)).

C. Kiuchi Anticipates The Asserted '504 And '211 Patent Claims.

For the '504 and '211 patents, the only limitations that VirnetX alleged were missing from Kiuchi were: (i) the "secure communication link" (and, particularly, its requirement of "direct communication"); and (ii) storing "a plurality of domain names and corresponding network addresses." A2332-2340(Jones); A8125. As to the former limitation, Kiuchi discloses "direct communication" between the client and the origin server as discussed above (p. 46).

As to the latter limitation, Kiuchi discloses "examining whether the requested server-side proxy is registered in the closed network" (A15009), which necessarily "involve[s] storing of the domain name and the IP address that corresponds to it" (A2073:11-14(Alexander); *see* A2072:13-19(Alexander)). The district court denied JMOL because "Jones contended that the Kiuchi reference was not enabling" (A53), but again the record does not support that conclusion. Jones merely testified that there was an error in the ***appendix*** to Kiuchi. A2337:2-20. He conceded, though, that the ***text*** of Kiuchi indicates that the client requests a URL located on the origin server (A2338:4-14), which involves storing domain names and corresponding IP addresses. A typographical error in a reference's appendix cannot undermine the disclosure in the reference itself, especially where, as here, an ordinarily skilled artisan would have understood the reference to be enabling. *See Application of Borst*, 345 F.2d 851, 853 n.2 (C.C.P.A. 1965).

IV. A NEW TRIAL IS WARRANTED BECAUSE THE DISTRICT COURT ERRED IN EXCLUDING EVIDENCE OF APPLE'S GOOD-FAITH BELIEF OF INVALIDITY.

To prove induced infringement, VirnetX was required to show that Apple knew or was willfully blind to the fact that its customers' use of Apple products would infringe valid patent claims. *See Global-Tech Appliances, Inc. v. SEB S.A.*, 131 S. Ct. 2060, 2068 (2011). To rebut that assertion, Apple sought to inform the jury that, after learning of VirnetX's allegations, Apple initiated reexaminations of the patents-in-suit that produced office actions rejecting all asserted claims—acts that showed Apple's reasonably-held belief that the patents were invalid. *See* A8002-8006.

The district court excluded the proffered evidence. A7993. As a result, the jury was left with a skewed and incomplete picture of Apple's reaction after learning of VirnetX’s patents. The court later opined that admitting evidence of the reexaminations would have been “highly prejudicial” and could have “improperly influenced the jury’s decision regarding validity.” A67.

The court’s ruling was legally erroneous. Evidence that presents a “good faith belief of invalidity” “should be considered by the fact-finder in determining whether an accused party knew ‘that the induced acts constitute patent infringement.’” *Commil USA, LLC v. Cisco Sys., Inc.*, 720 F.3d 1361, 1368-1369 (Fed. Cir. 2013). Apple did not believe it was inducing others to infringe, and the reexamination files show that Apple had “a good-faith belief that the patent[s are] not valid,” under which circumstances “it can hardly be said that the alleged inducer intended to induce infringement.” *Id.* at 1368. The error was particularly harmful here, as the court allowed VirnetX to introduce evidence that individual Apple engineers did not read VirnetX’s patents and to argue that Apple ***as a company*** was thus willfully blind to infringement. A1735:18-1760:12 (deposition testimony of eight Apple engineers); A2528:5-8, A2584:22-2585:2 (references to deposition testimony in VirnetX’s summation). In fact, Apple ***as a company*** initiated reexamination proceedings in which, at the time of trial and as of the filing of this brief, all asserted claims stood rejected, but the jury was precluded from hearing that.[9]

Because Apple's proffered evidence "should be considered by the fact-finder" in its consideration of induced infringement (*Commil*, 720 F.3d at 1369), this Court should reverse the order excluding the reexamination evidence. *See Davidson Oil Country Supply, Inc. v. Klockner, Inc.*, 908 F.2d 1238, 1245 (5th Cir. 1990) (ordering new trial where "excluded evidence was clearly relevant" and "its exclusion seriously hindered" the presentation of the case). Because the district court (over Apple's objection) refused to ask the jury to make separate inducement and direct infringement findings (A2473:18-2474:5), and the jury was instructed to consider inducement as to all asserted claims (A8023), a new trial is required on all claims.

V. THE DAMAGES AWARD SHOULD BE REVERSED OR REMANDED FOR A NEW TRIAL.

A. The District Court Legally Erred In Instructing The Jury That It Could Base A Royalty On A Product's Entire Market Value Where The Claimed Invention Does *Not* Drive Consumer Demand.

"'[I]n every case,'" a patentee must either (i) "'give evidence tending to separate or apportion the defendant's profits between the patented feature and the

[9] The court cited *Callaway Golf Co. v. Acushnet Co.*, 576 F.3d 1331 (Fed. Cir. 2009), but there evidence of reexamination proceedings was intended not to show a ***good-faith belief*** of invalidity, but as ***direct evidence*** on "the question of obviousness." *Id.* at 1343.

unpatented feature,'" or (ii) "show that 'the entire value of the whole machine, as a marketable article, is properly and legally attributable to the patented feature.'" *Uniloc USA, Inc. v. Microsoft Corp.*, 632 F.3d 1292, 1318 (Fed. Cir. 2011) (quoting *Garretson v. Clark*, 111 U.S. 120, 121 (1884)). The second option, the entire market value rule, is a "narrow exception" to the general rule for calculating reasonable royalty damages, *LaserDynamics, Inc. v. Quanta Computer, Inc.*, 694 F.3d 51, 67 (Fed. Cir. 2012), available "***only where*** the patented feature creates the 'basis for customer demand' or 'substantially creates the value of the [unclaimed] component parts,'" *id.* (quoting *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1336 (Fed. Cir. 2009)).

The district court erroneously instructed the jury that there was a ***second*** situation in which damages could be based on the products' entire market value:

> In determining the royalty base, you should not use the value of the entire apparatus or product ***unless either***: (1) the patented feature creates the basis for the customer's demand for the product, or the patented feature substantially creates the value of the other component parts of the product; ***or (2) the product in question constitutes the smallest saleable unit containing the patented feature.***

A2515:24-2516:9. The court later confirmed that it had created "***another alternative*** when the jury could consider the entire value of a product." A64.

The district court did not attempt to reconcile its new "alternative" with *Garretson* or *Uniloc*. The court's only reasoning was that "it is foreseeable that an entire product will be required to practice an invention." A50. But the authority it cited—a passage from *Rite-Hite Corp. v. Kelley Co.*, 56 F.3d 1538, 1550 (Fed. Cir. 1995) (en banc)—discussed (and rejected) an attempt to recover damages on "items that have essentially no functional relationship to the patented invention," noting that the law ***never*** permitted recovery for unpatented features that were not "analogous to components of a single assembly or … parts of a complete machine." *Id.* But even in the case of a "complete machine," recovery based on the value of unpatented components requires satisfying the "entire market value rule," which *Rite-Hite* discusses at length. *Id.* at 1549-1551. Nothing in *Rite-Hite* suggests the district court's "alternative" approach, whereby a patentee may base damages on a product's entire market value ***without*** satisfying the entire market value rule, simply by showing that practicing the claimed invention requires a larger "entire product" that includes other features. A50. If that were the law, many cases rejecting reasonable royalty theories would have been wrongly decided. *See, e.g.*, *Lucent*, 580 F.3d at 1337-1338 (vacating damages award where plaintiff used revenues from sales of Microsoft Outlook, the smallest saleable unit, as a royalty base when the patented "date-picker tool" was "a very small component of a much larger software program" and did not drive demand); *Uniloc*, 632 F.3d at 1318-1321 (applying similar analysis and rejecting the use of Outlook revenues as a royalty base for infringement based on the "Product Activation" feature, which did not drive demand).

VirnetX cited *LaserDynamics* below, but that decision supports reversal: this Court found that calculation of a royalty based on a laptop's entire market value was ***impermissible*** precisely because, as here, the plaintiff presented no evidence that the patent-practicing disk drives drove demand for the laptops. 694 F.3d at 68. Indeed, the Court held that there is "no reason to establish a necessity-based exception to the entire market value rule." *Id.* at 69-70.

While *LaserDynamics* contained *dicta* stating that a patentee must "generally" base royalties on the "smallest salable patent-practicing unit" (694 F.3d at 67), this Court has never suggested that a patentee may do so even when that "unit" includes numerous unclaimed features. On the contrary, the Court used that phrase in ***rejecting*** efforts by patentees to base royalties on the entire value of complex products. *See id.* (royalties are "generally" based "***not on the entire product, but instead*** on the 'smallest salable patent practicing unit'"). Resort to a "smallest salable unit" does not excuse a patentee from the Supreme Court's requirement of "***in every case*** giv[ing] evidence tending to separate or apportion the defendant's profits and the patentee's damages between the patented feature and the unpatented feature." *Garretson*, 111 U.S. at 121; *see also LaserDynamics*, 694 F.3d at 67 (a patentee can receive damages "as a percentage of revenues or profits attributable to the entire product" "***[i]f it can be shown*** that the patented feature drives demand for an entire multi-component product"); *Network Prot.*

Sciences, LLC v. Fortinet, Inc., 2013 WL 5402089, at *6-8 (N.D. Cal. Sept. 26, 2013); *Dynetix Design Solutions, Inc. v. Synopsys, Inc.*, 2013 WL 4538210, at *3 (N.D. Cal. Aug. 22, 2013) ("*LaserDynamics* supports the premise that an apportionment is required even where there the accused product is the smallest salable unit."). Apportionment is essential in cases like this, to ensure that the royalty base does "not overreach and encompass components not covered by the patent." *LaserDynamics*, 694 F.3d at 70; *see also Garretson*, 111 U.S. at 121 ("[T]he patentee must show in what particulars his improvement has added to the usefulness of the machine or contrivance.").[10]

The erroneous instruction was unquestionably prejudicial. VirnetX specifically referenced it in closing. A2543:6-2544:14 (arguing that Weinstein "used the smallest saleable price ***that Judge Davis has described to you in his instruction***"). Such an argument is precisely what the entire market value rule is meant to preclude. *LaserDynamics*, 694 F.3d at 67 (allowing a patentee to use the accused products' entire market value as the royalty base, even though the infringing feature is only one part of a multi-component product, creates "a considerable risk that the patentee will be improperly compensated for non-infringing components of that product"); *Uniloc*, 632 F.3d at 1320 (permitting

[10] To the extent the Court reads *LaserDynamics* as VirnetX does, Apple respectfully submits that, in that regard, it is inconsistent with *Garretson*, *Rite-Hite*, and *Uniloc*, and should not be followed.

introduction of the alleged infringer's entire profits "skew[s] the damages horizon for the jury"). Accordingly, the Court should order a new trial.[11]

B. VirnetX's Damages Testimony Cannot Support A Verdict And Should Have Been Excluded.

1. VirnetX's excessive royalty base was legally unjustified.

VirnetX had two options in determining a royalty base: it could have (i) apportioned profits between patented and unpatented features, or (ii) "show[n] that 'the entire value of the whole machine [is] attributable to the patented feature.'" *Uniloc*, 632 F.3d at 1318 (quoting *Garretson*, 111 U.S. at 121). VirnetX did neither. Meanwhile, the district court faulted Apple for supposedly not advancing a "credible alternative" royalty base (A47), even though "the patentee bears the burden of proving damages." *WhitServe, LLC v. Computer Packages, Inc.*, 694 F.3d 10, 26 (Fed. Cir. 2012).

VirnetX's expert Weinstein indisputably based his calculations on the cost of entire iOS products—ranging from $199 for iPod touch to $649 for iPhone 4S. A1616:10-20; A1621:20-1622:2; A1622:19-24; 1673:16-19; 1698:5-8. While he used the base-model price for each mobile product, thus excluding revenue for additional memory, that does not change the fact that he employed the entire

[11] Because the jury instruction was legally erroneous, the judgment cannot be affirmed on the speculation that the jury could have based its verdict on another theory. *See i4i Ltd. P'ship v. Microsoft Corp.*, 598 F.3d 831, 849 (Fed. Cir. 2010) (citing *Walther v. Lone Star Gas Co.*, 952 F.2d 119, 126 (5th Cir. 1992)), *aff'd*, 131 S. Ct. 2238 (2011).

market value of the base model products and, accordingly, credited VirnetX with the value of hundreds of features not covered by the patents. *See LaserDynamics*, 694 F.3d at 68 (a "royalty expressly calculated as a percentage of the entire value of the product … by definition, is an application of the entire market value rule"). This is true regardless of Weinstein's calling this figure "apportioned revenue." A1621:9-17.

Because Weinstein's analysis began with the value of entire products, he was required to show that VirnetX's technology drove demand for those products. *Uniloc*, 632 F.3d at 1318. But Weinstein ***conceded*** that neither FaceTime nor VPN On Demand—much less the minor aspects of those features that VirnetX claimed were infringing—drove demand for Apple's products. When asked whether he meant "to offer the opinion that either … FaceTime or VPN On Demand [was] the basis for demand for the Apple products," he answered, "No," and asserted that the features merely "***contribute to*** the sales of … the products."[12] A1622:21-1663:3. But merely "contributing" to sales is not sufficient; the proper inquiry is "what motivates consumers to buy [the defendant's product] in the first place." *LaserDynamics*, 694 F.3d at 68 ("It is not enough to merely show that the

[12] Even if Weinstein ***had*** testified that FaceTime and VPN On Demand created the demand for Apple's products, that would not be equivalent to testifying that ***VirnetX's patented technology*** drove demand. Named inventor Short admitted that there is a "bunch of stuff in FaceTime and VPN On Demand that's not covered by VirnetX's patents." A1143:5-1146:12.

[patented technology] is valuable, important, or even essential to the use of the [defendant's product]."). Apple's mobile devices undisputedly contain numerous important features that VirnetX did not invent. A1143:1-1144:18. Weinstein did not even attempt to show that VirnetX's technology was "what motivates customers to buy [Apple's products] in the first place," thus failing to meet the "higher degree of proof" required by the entire market value rule. *LaserDynamics*, 694 F.3d at 68.

Weinstein's attempt to draw a temporal connection between Apple's adoption of the accused features and increased sales of Apple's products was also improper. A1603:16-1604:12. Apple added numerous other features and improvements during the same period, and also broadened its market by expanding to additional wireless carriers. A1714:18-1715:19. The mere fact that Apple increased its sales during this period does not demonstrate that the increase was due to the claimed technology. *See U.S. Steel Grp. v. United States*, 96 F.3d 1352, 1358 (Fed. Cir. 1996) ("[T]o claim that the temporal link between these events *proves* that they are causally related is simply to repeat the ancient fallacy: *post hoc ergo propter hoc*." (italics in original)).

Weinstein's use of survey data likewise fails to demonstrate that VirnetX's claimed inventions drove demand for Apple's products. Weinstein did not even attempt to show that VPN On Demand drove demand. A1675:24-1676:3. His discussion of surveys mentioning FaceTime failed to account for the fact that some FaceTime calls are placed through admittedly non-infringing relay servers. A1445:7-10. Thus, even if ***FaceTime*** drove demand, ***VirnetX's technology*** would not. And Weinstein's cited survey data shows at most that [redacted].[13] That a particular feature, of which only a small part could even arguably be attributed to VirnetX's technology, served as ***one reason among many*** for a minority of customers purchasing one of many products cannot prove that VirnetX's technology motivated customers to buy all accused iOS products.[14]

Even under Weinstein's own analysis, there was no need to use the entire value of iOS products as a base. Where the same exact technology in Mac computers was concerned, he used a royalty base of $29—the price of a Mac software upgrade that included FaceTime. A1619:15-21. However, while Weinstein believed that VirnetX would accept $0.29 to allow Apple to make FaceTime available in a Mac, he believed VirnetX would charge and Apple would

[13] The survey that Weinstein relied upon showed that, of the [redacted] 18% listed FaceTime as the most desired feature; 18% of [redacted] A1640:23-1641:2; A15292.

[14] Weinstein's application of the iPod touch survey results to other iOS products is especially unreliable, as those products contain different features and benefits. A1806:14-1807:9. Surveys regarding other products differed widely; for example, only 3% of iPhone customers listed FaceTime as the most desired feature. A1712:22-1713:11.

pay $6.49 to make the same feature available in a mobile device—an illogical situation that Weinstein acknowledged but blithely accepted. A1672:12-1673:23, A1698:1-1699:18.

Weinstein stated that he did not use the $29 Mac software upgrade price to calculate a royalty base for iOS devices because Apple does not charge for iOS software upgrades. A1663:11-15; A1673:24-1674:7. But the fact that Apple gives customers iOS upgrades for free does not mean that the upgrades cannot be valued or are not "saleable."[15] "[E]stimates in the damages context" are permissible, including for hypothetical revenue in apportionment or smallest saleable unit contexts. *Cornell Univ. v. Hewlett-Packard Co.*, 609 F. Supp. 2d 279, 290 (Rader, C.J., sitting by designation), *amended*, 2009 WL 1405208 (N.D.N.Y. May 15, 2009). Indeed, Weinstein ***was*** able to estimate the value of VirnetX's supposed contribution to iOS products in his "Nash bargaining solution" analysis, where he purported to calculate the incremental revenues due to FaceTime at ***$15 per iOS device***, well below the royalty base used in his principal theory. A1634:8-1636:10.

VirnetX's royalty base thus neither satisfied the entire market value rule nor properly apportioned the value due to the claimed invention. Weinstein's royalty base testimony should have been excluded under *Daubert* and cannot support a

[15] While Weinstein claimed that the software upgrade was "included with … the price of the product," he made no attempt to determine how much of the product price was attributable to the update. A1673:24-1674:4.

verdict. *See Uniloc*, 632 F.3d at 1321 (affirming grant of new trial due to "Uniloc's violation of the entire market value rule").

2. VirnetX's royalty rate evidence should have been excluded.

Although a royalty rate may be partly based on "royalties received by the patentee for the licensing of the patent in suit, proving or tending to prove an established royalty" (*Georgia-Pacific Corp. v. U.S. Plywood Corp.*, 318 F. Supp. 1116, 1120 (S.D.N.Y. 1970)), a patentee may not rely on licenses that have "no relationship to the claimed invention" or are not "commensurate with" the accused infringement. *ResQNet.com, Inc. v. Lansa, Inc.*, 594 F.3d 860, 868, 872 (Fed. Cir. 2010). The licenses VirnetX presented failed these requirements.

"By its terms, [*Georgia-Pacific*'s first] factor considers only past and present licenses ***to the actual patent and the actual claims in litigation***." *ResQNet*, 594 F.3d at 869; *see* 7-20 *Chisum on Patents* § 20.06[2] (2012) (an "established royalty" requires proof of a royalty rate "paid or secured before the infringement complained of" and "for comparable rights or activity under the patent").[16] Two of the licenses (SAIC/SafeNet and In-Q-Tel) predated the patents-in-suit. A1678:2-1679:12. The In-Q-Tel license predated even the conception of the claimed

[16] Weinstein's reliance on VirnetX's supposed "licensing policy" of issuing licenses with royalty rates of 1%-2% is flawed, as he admitted that no license had been issued under the policy at the time of the hypothetical negotiation and that the licensing policy was merely what VirnetX "hoped for." A1595:6-19; A1652:23-1653:13.

inventions (A1678:2-8), and the SAIC/SafeNet license was not a patent license but a software license, which Weinstein admitted "can have higher rates" (A1678:9-1679:3). Basing a royalty rate on these licenses "unrelated to the claimed invention does not support compensation for infringement but punishes beyond the reach of the statute." *ResQNet*, 594 F.3d at 869.

The Microsoft, Aastra, Mitel, and NEC licenses were likewise inapposite. The Aastra, Mitel, and NEC licenses were entered into after the 2009 hypothetical negotiation between Apple and VirnetX, when VirnetX had a much stronger financial picture and bargaining position than the "financial dire straits" it faced in 2009. A1118; A1649:20-1650:9; 1651:5-12; *see, e.g.*, *LaserDynamics*, 694 F.3d at 78 (settlement agreement post-dating hypothetical negotiation by three years was "in many ways not relevant to the hypothetical negotiation" due to the "changing technological and financial landscape"); *Odetics, Inc. v. Storage Tech. Corp.*, 185 F.3d 1259, 1276-1277 (Fed. Cir. 1999) (licenses dated 4-5 years after the hypothetical negotiation were irrelevant). And these licenses did not involve a potential royalty stream remotely approaching that claimed by VirnetX in this case—each involved less than $1 million. A1694:5-1697:12. The Microsoft license gave that company rights to 68 VirnetX patents, where Apple would only have sought rights under four, and thus created a much broader license than Apple would have needed here.[17] A1682:1-1683:2. Further, the Microsoft license was a lump sum with an effective royalty rate calculated by Weinstein to be 0.24%, one quarter of the lowest rate VirnetX claimed it would accept. A1680:4-1681:1, A1683:3-18. Apple's expert Christopher Vellturo, using a different method, calculated the effective royalty rate to be half of ***that***—0.115%. A1684:18-21; A2287:3-11.

Thus, contrary to the district court's statement, these licenses cannot "help demonstrate what entities would be willing to pay for the technology at issue." A48. Rather, they are not "commensurate with" the accused infringement and are "speculative evidence [that] violates the statutory requirement that damages … 'be adequate to compensate for the infringement.'" *ResQNet*, 594 F.3d at 872-873.

3. VirnetX's "Nash bargaining solution" theory also should have been excluded.

Even if Weinstein's separate Nash theory were proper, it could not avoid the need for a new trial, because it applied only to the two patents asserted against FaceTime; for VPN On Demand, Weinstein relied exclusively on his first theory, which was inadmissible as explained above (at pp. 56-63). Because the verdict form asked the jury to render a single damages figure (A241), notwithstanding Apple's objection that damages be separated by accused feature (A2474:6-

[17] The Aastra, Mitel, and NEC licenses, which gave those companies rights to 16, 17, and 17 patents respectively (A2270:17-25), are also not "commensurate with" the accused infringement. *ResQNet*, 594 F.3d at 872.

2475:10), there is no way to separate out improperly-awarded VPN On Demand damages, even if the Court were to hypothesize that damages for FaceTime could be upheld using the Nash theory. A new trial is therefore invariably required, even if the Nash theory were permissible.

It is not permissible, however. As other courts have ruled—including one that excluded ***Weinstein's own*** Nash-related testimony—the Nash theory relies on assumptions that "idealize the bargaining problem" and may not align with real-world conditions. *Oracle Am., Inc. v. Google Inc.*, 798 F. Supp. 2d 1111, 1119 (N.D. Cal. 2011) ("The Nash bargaining solution has never been approved by a judge to calculate reasonable royalties in litigation, at least in the face of objection."); *Suffolk Techs. LLC v. AOL Inc.*, 2013 U.S. Dist. LEXIS 64630, at *5-6 (E.D. Va. Apr. 12, 2013) (excluding Weinstein's Nash testimony as "not tied to the facts of the case").

Indeed, Weinstein's Nash theory is indistinguishable from the "25 percent rule of thumb" this Court rejected in *Uniloc*, which "suggest[ed] that the licensee would pay a royalty rate equivalent to 25 per cent of its expected profits." 632 F.3d at 1312. Weinstein's Nash theory similarly began by assuming that negotiating parties would each take 50% of incremental profits. *See* Choi & Weinstein, *An Analytical Solution to Reasonable Royalty Rate Calculations*, 41 IDEA 49, 55 (2001). That 50% starting point had no "basis in fact to associate royalty rates used in past licenses to the particular hypothetical negotiation at issue in the case." *Uniloc*, 632 F.3d at 1317; *see Oracle*, 798 F. Supp. 2d at 1119 ("There is no anchor for [the Nash theory's] fifty-percent assumption in the record of actual transactions."). Weinstein made no effort to ground the 50/50 split in the facts of the case. A1633:11-17. Nor does the fact that he purported to adjust the 50/50 split based on "the relative bargaining power of the two entities" (A1632:12-17) remove its arbitrariness; "[b]eginning from a fundamentally flawed premise and adjusting it based on legitimate considerations specific to the facts of the case nevertheless results in a fundamentally flawed conclusion." *Uniloc*, 632 F.3d at 1317; *see Suffolk*, 2013 U.S. Dist. LEXIS 64630, at *5 n.2 (noting this similarity between the Nash theory and the "25 percent" rule).

Weinstein's Nash theory accordingly "invite[d] a miscarriage of justice by clothing a fifty-percent assumption in an impenetrable facade of mathematics." *Oracle*, 798 F. Supp. 2d at 1120. Indeed, the adjustment Weinstein made—from 50/50 to 45/55—was itself arbitrary. His only explanation for that minor adjustment was that "Apple would have [had] additional bargaining power over VirnetX back in … 2009." A1708:11-1709:4. Weinstein never explained why he chose a 10% variation and not a much greater figure. His 45/55 split was thus "plucked out of thin air" with a "complete lack of economic analysis." *LaserDynamics*, 694 F.3d at 69.[18]

Furthermore, even if a "Nash bargaining solution" approach might be admissible in some circumstances, Weinstein clearly misapplied it here. Weinstein announced that a Nash analysis begins with "the incremental or additional profits that are associated with ***the use of the patented technology***." A1630:12-14. However, he began with the alleged incremental revenue from the addition of the ***front-facing camera*** to certain Apple products (A1710:4-11), despite the named inventor's concession that the "cameras that take video for FaceTime" "don't have anything to do with" VirnetX's patents (A1143:16-1144:18), and Weinstein himself admitted that the camera serves functions in addition to FaceTime (A1710:9-11, 22-25). Although he then claimed that $1.3 billion in profit was attributable to FaceTime, given that VirnetX did not even contend that its patents cover all of ***FaceTime***, but at most only one way to connect FaceTime calls without a relay server (A1445:3-10), Weinstein had no basis for ascribing $1.3 billion in profit to VirnetX's unrelated patents. Weinstein's Nash analysis was

[18] Weinstein coincidentally arrived at the same arbitrary 45/55 split for a completely separate 2005 hypothetical negotiation between SAIC and Cisco. A7987.

thus unreliable in theory and applied unreliably on its own arbitrary terms. It should have been excluded.[19]

4. Weinstein's application of the "entire market value rule" fell far short of that rule's requirements.

Weinstein's final damages theory, which he called an application of the entire market value rule, was equally improper. He asserted that his theory was a "method of measuring the contribution of the patents-in-suit to the sales of the product." A1638:20-1639:6. However, the entire market value rule is not available for ***any*** measure of "contribution … to the sales of the product"; it is available only if the patented invention "motivates consumers to buy [the product] in the first place." *LaserDynamics*, 694 F.3d at 68. Weinstein did not come close to showing that VirnetX's patented technology drove demand for Apple's products; in fact, he conceded the contrary. A1662:21-1663:3; *see supra* pp. 57-59.

Even had Weinstein possessed sufficient information to support the use of the entire market value rule, he rendered it impermissibly speculative. Instead of calculating a royalty base, he used the portion of profits he assumed Apple would not have received "but for" customers' interest in FaceTime; instead of using a

[19] That the Nash theory was presented in addition to other damages theories does not change this. Even where an improper damages theory is presented as "only a check," it taints the jury's consideration and warrants a new trial. *Uniloc*, 632 F.3d at 1321.

royalty rate, he multiplied those profits by the arbitrary 45% figure from his Nash theory. A1641:11-1643:5. That alone warrants the theory's exclusion. *See Uniloc*, 632 F.3d at 1317. In all, Weinstein's analysis fell far short of the "higher degree of proof that must exist to support an entire market value rule theory." *LaserDynamics*, 694 F.3d at 68.

* * *

The district court should have excluded Weinstein's testimony in its entirety. Because that would leave VirnetX with ***no*** proper damages evidence, this Court may order judgment for Apple. *See Weisgram v. Marley Co.*, 528 U.S. 440, 457 (2000). Alternatively, the Court should order a new trial.[20]

[20] If any one of VirnetX's damages theories is held legally inadequate, a full new damages trial is necessary. *See Uniloc*, 632 F.3d at 1295.

CONCLUSION

The district court's judgment with respect to infringement, validity, and damages should be reversed or, alternatively, remanded for a new trial.

Dated: October 17, 2013

Respectfully submitted,

/s/ William F. Lee
WILLIAM F. LEE
MARK C. FLEMING
LAUREN B. FLETCHER
REBECCA BACT
WILMER CUTLER PICKERING
HALE AND DORR LLP
60 State Street
Boston, MA 02109
(617) 526-6000

JONATHAN G. CEDARBAUM
BRITTANY BLUEITT AMADI
LEAH LITMAN
WILMER CUTLER PICKERING
HALE AND DORR LLP
1875 Pennsylvania Ave., NW
Washington, DC 20006
(202) 663-6000

DANNY L. WILLIAMS
WILLIAMS, MORGAN & AMERSON, P.C.
10333 Richmond, Suite 1100
Houston, TX 77042
(713) 934-7000

Attorneys for Defendant-Appellant
Apple Inc.

CERTIFICATE OF SERVICE

I hereby certify that I filed the foregoing Brief for Defendant-Appellant Apple Inc. with the Clerk of the United States Court of Appeals for the Federal Circuit via the CM/ECF system this 17th day of October, 2013, and served a copy on counsel of record by the CM/ECF system and by electronic mail to the parties on the service list below.

J. Michael Jakes
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, LLP
901 New York Avenue, N.W.
Washington, DC 20001
(202) 408-4000
mike.jakes@finnegan.com

Donald Urrabazo
URRABAZO LAW, P.C.
2029 Century Park East
Suite 1400
Los Angeles, CA
(310) 388-9096
durrabazo@ulawpc.com

Dated: October 17, 2013

<u>/s/ William F. Lee</u>
WILLIAM F. LEE
WILMER CUTLER PICKERING
HALE AND DORR LLP
60 State Street
Boston, MA 02109
(617) 526-6000

CERTIFICATE OF COMPLIANCE

Pursuant to Fed. R. App. P. 32(a)(7)(C), the undersigned hereby certifies that this brief complies with the type-volume limitation of Fed. R. App. P. 32(a)(7)(B)(i).

1. Exclusive of the exempted portions of the brief, as provided in Fed. R. App. P. 32(a)(7)(B), the brief contains 13,988 words.

2. The brief has been prepared in proportionally spaced typeface using Microsoft Word 2010 in 14 point Times New Roman font. As permitted by Fed. R. App. P. 32(a)(7)(B), the undersigned has relied upon the word count feature of this word processing system in preparing this certificate.

Dated: October 17, 2013

/s/ William F. Lee
WILLIAM F. LEE
WILMER CUTLER PICKERING
HALE AND DORR LLP
60 State Street
Boston, MA 02109
(617) 526-6000

ADDENDUM

TABLE OF CONTENTS

TAB 1

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

	§	
VIRNETX INC.,	§	
	§	
Plaintiff,	§	
	§	
vs.	§	**CASE NO. 6:10-CV-417**
	§	
CISCO SYSTEMS, INC., et al.,	§	
	§	
Defendants.	§	
	§	

MEMORANDUM OPINION AND ORDER

This Memorandum Opinion construes the disputed claim terms in U.S. Patent Nos. 6,502,135 (“the ‘135 Patent”), 6,839,759 (“the ‘759 Patent”), 7,188,180 (“the ‘180 Patent”), 7,418,504 (“the ‘504 Patent”), 7,490,151 (“the ‘151 Patent”), and 7,921,211 (“the ‘211 Patent”).

Further, as stated at the *Markman* hearing and agreed by the parties, the Court **ORDERS** that VirnetX Inc.’s Motion to Compel from Apple a Complete Response to VirnetX’s Eighth Common Interrogatory (Docket No. 179) is **DENIED AS MOOT**.

BACKGROUND

VirnetX Inc. (“VirnetX”) asserts all six patents-in-suit against Aastra Technologies Ltd.; Aastra USA, Inc.; Apple Inc.; Cisco Systems, Inc.; NEC Corporation; and NEC Corporation of America (collectively “Defendants”). The ‘135 Patent discloses a method of transparently creating a virtual private network (“VPN”) between a client computer and a target computer. The ‘759 Patent discloses a method for establishing a VPN without a user entering user identification information. The ‘180 Patent discloses a method of establishing a secure communication link between two computers. The ‘504 and ‘211 Patents disclose a secure domain name service. The

'151 Patent discloses a domain name service capable of handling both standard and non-standard domain name service queries.

The patents-in-suit are all related; Application No. 09/504,783 ("the '783 Application") is an ancestor application for every patent-in-suit. The '135 Patent issued on December 31, 2002, from the '783 Application. The '151 Patent issued from a division of the '783 Application. The '180 Patent issued from a division of a continuation-in-part of the '783 Application. Both the '759 and '504 Patents issued from a continuation of a continuation-in-part of the '783 Application. Finally, the '211 Patent is a continuation of the application that resulted in the '504 patent.

The Court has already construed many of the terms at issue in a previous case that involved the '135, '759, and '180 Patents. *See VirnetX, Inc. v. Microsoft Corp.*, 2009 U.S. Dist. LEXIS 65667, No. 6:07cv80 (E.D. Tex. July 30, 2009) ("*Microsoft*").

APPLICABLE LAW

"It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (quoting *Innova/Pure Water Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004)). In claim construction, courts examine the patent's intrinsic evidence to define the patented invention's scope. *See id.*; *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 861 (Fed. Cir. 2004); *Bell Atl. Network Servs., Inc. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1267 (Fed. Cir. 2001). This intrinsic evidence includes the claims themselves, the specification, and the prosecution history. *See Phillips*, 415 F.3d at 1314; *C.R. Bard, Inc.*, 388 F.3d at 861. Courts give claim terms their ordinary and accustomed meaning as understood by one of ordinary skill in the art at the time of the invention in the context of the entire patent. *Phillips*, 415 F.3d at 1312–13; *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

The claims themselves provide substantial guidance in determining the meaning of particular claim terms. *Phillips*, 415 F.3d at 1314. First, a term's context in the asserted claim can be very instructive. *Id.* Other asserted or unasserted claims can also aid in determining the claim's meaning because claim terms are typically used consistently throughout the patent. *Id.* Differences among the claim terms can also assist in understanding a term's meaning. *Id.* For example, when a dependent claim adds a limitation to an independent claim, it is presumed that the independent claim does not include the limitation. *Id.* at 1314–15.

"[C]laims 'must be read in view of the specification, of which they are a part.'" *Id.* (quoting *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (en banc)). "[T]he specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id.* (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)); *see also Teleflex, Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002). This is true because a patentee may define his own terms, give a claim term a different meaning than the term would otherwise possess, or disclaim or disavow the claim scope. *Phillips*, 415 F.3d at 1316. In these situations, the inventor's lexicography governs. *Id.* Also, the specification may resolve ambiguous claim terms "where the ordinary and accustomed meaning of the words used in the claims lack sufficient clarity to permit the scope of the claim to be ascertained from the words alone." *Teleflex, Inc.*, 299 F.3d at 1325. But, "'[a]lthough the specification may aid the court in interpreting the meaning of disputed claim language, particular embodiments and examples appearing in the specification will not generally be read into the claims.'" *Comark Commc'ns,* *Inc. v. Harris Corp.*, 156 F.3d 1182, 1187 (Fed. Cir. 1998) (quoting *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1571 (Fed. Cir. 1988)); *see also Phillips*, 415 F.3d at 1323. The prosecution history is another tool to supply the proper context for claim construction because a patent applicant may also define a term in prosecuting the patent. *Home Diagnostics, Inc., v. Lifescan, Inc.*, 381 F.3d 1352, 1356 (Fed. Cir. 2004) ("As in the case of the specification, a patent applicant may define a term in prosecuting a patent.").

Although extrinsic evidence can be useful, it is "'less significant than the intrinsic record in determining the legally operative meaning of claim language.'" *Phillips*, 415 F.3d at 1317 (quoting *C.R. Bard, Inc.*, 388 F.3d at 862). Technical dictionaries and treatises may help a court understand the underlying technology and the manner in which one skilled in the art might use claim terms, but technical dictionaries and treatises may provide definitions that are too broad or may not be indicative of how the term is used in the patent. *Id.* at 1318. Similarly, expert testimony may aid a court in understanding the underlying technology and determining the particular meaning of a term in the pertinent field, but an expert's conclusory, unsupported assertions as to a term's definition is entirely unhelpful to a court. *Id.* Generally, extrinsic evidence is "less reliable than the patent and its prosecution history in determining how to read claim terms." *Id.*

Defendants also contend that some claims at issue are invalid for indefiniteness. A claim is invalid under 35 U.S.C. § 112 ¶ 2 if it fails to particularly point out and distinctly claim the subject matter that the applicant regards as the invention. The party seeking to invalidate a claim under 35 U.S.C. § 112 ¶ 2 as indefinite must show by clear and convincing evidence that one skilled in the art would not understand the scope of the claim when read in light of the specification. *Intellectual Prop. Dev., Inc. v. UA-Columbia Cablevision of Westchester, Inc.*, 336 F.3d 1308, 1319 (Fed. Cir. 2003).

LEVEL OF ORDINARY SKILL IN THE ART

The parties agree that a person of ordinary skill in the art would have a master's degree in computer science or computer engineering and approximately two years of experience in computer networking and computer network security.

CLAIM TERMS

virtual private network

VirnetX proposes "a network of computers which privately communicate with each other by encrypting traffic on insecure communication paths between the computers." Defendants propose the following emphasized additions: "a network of computers which privately ***and directly*** communicate with each other by encrypting traffic on insecure communication paths between the computers ***where the communication is both secure and anonymous***."

secure and anonymous

VirnetX proposes the same construction adopted by this Court in *Microsoft*. *See Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *8. Defendants seek to explicitly include the "secure and anonymous" language that was implicitly included in the Court's *Microsoft* construction. *See id.* at *16 ("[T]he Court construes 'virtual private network' as requiring both data security and anonymity."). Just as in *Microsoft*, the parties here dispute whether a virtual private network requires anonymity, and the Court hereby incorporates by reference its reasoning in *Microsoft*. *See id.* at *14–17. For the same reasons stated in *Microsoft*, the Court finds that a virtual private network requires both data security and anonymity. For clarity, this language is now explicitly included in the Court's construction of "virtual private network."

directly

Defendants propose that communication within a virtual private network is "direct" based on arguments that VirnetX made to the United States Patent and Trademark Office ("PTO") to overcome rejections based on the Aventail reference during reexamination of the '135 Patent.[1] VirnetX provided three reasons that Aventail did not disclose a virtual private network:

> First, Aventail has not been shown to demonstrate that computers connected via the Aventail system are able to communicate with each other as though they were on the same network. . . .
> . . .
> Second, according to Aventail, Aventail Connect's fundamental operation is incompatible with users transmitting data that is sensitive to network information. . . .
> Third, Aventail has not been shown to disclose a VPN because computers connected according to Aventail do not communicate directly with each other.

Docket No. 182 Attach. 16, at 5–7. Defendants argue that VirnetX's third distinction warrants a finding that communication over a virtual private network must be direct.

VirnetX argues that its statements during reexamination are not a clear disavowal of claim scope. Rather, VirnetX contends that it "overcame Aventail on the ground that Aventail did not teach a VPN at all." Docket No. 173, at 8. However, the statements made by VirnetX—particularly points one and three—reveal that the *reason* Aventail did not disclose a VPN was because it did not permit direct communication between the source and target computers.

VirnetX further argues that it did not clearly disavow claim scope regarding any one of the three distinctions between Aventail and a VPN. For support, VirnetX relies on *Momentus Golf, Inc. v. Swingrite Golf Corp.*, 187 Fed. App'x 981 (Fed. Cir. 2006), which involved a patent directed to a golf club swing aide. During prosecution of the *Momentus Golf* patent, the applicants stated: "A hollow device having 10–25% club head weight cannot meet the

[1] The Aventail reference involved a means of secure communication between two clients via an intermediary SOCKS server.

requirement in applicant's claims that the center of gravity of the trainer be substantially at the center of a solid round stock." *Momentus Golf*, 187 Fed. App'x at 984 (quoting prosecution history). The district court held that this statement presented a clear disavowal of golf trainers with 10–25% club head weight because they would not meet the center of gravity requirement. *Id.* at 982. The Federal Circuit agreed that the district court's interpretation was a fathomable one. *Id.* at 983–84. However, it reversed the district court because another interpretation was also reasonable and still supported the applicant's distinguishing arguments—that the statement only clearly disavowed *hollow clubs* with 10–25% club head weight. *Id.* at 984 (emphasis added). The Federal Circuit held that the statement could reasonably be interpreted to disavow (1) clubs with 10–25% club head weight or (2) hollow clubs with 10–25% club head weight. In light of the competing interpretations, the Federal Circuit determined that there was only a disclaimer of the more narrow interpretation.

The instant case does not present such an ambiguous statement. VirnetX stated that "Aventail has not been shown to disclose the VPN . . . for at least three reasons." Docket No. 182 Attach. 16, at 5. VirnetX then proceeded to independently present and discuss each of the three distinct reasons that Aventail did not disclose the claimed VPN. *See* Docket No. 182 Attach. 16, at 5–6 (discussing the first reason); *id.* at 6–7 (discussing the second reason); *id.* at 7 (discussing the third reason). In *Momentus Golf*, the applicant combined two potential distinctions in a single sentence, creating ambiguity as to whether the distinctions were independent or intertwined. Here, VirnetX expressly stated that there were three bases for distinction. Each of these reasons, alone, served to distinguish the claimed VPN from the Aventail reference. *See Andersen Corp. v. Fiber Composites, LLC*, 474 F.3d 1361, 1374 (Fed. Cir. 2007) ("An applicant's invocation of multiple grounds for distinguishing a prior art reference does not immunize each of them from being used to construe the claim language.”). Accordingly, the Court finds that the claimed “virtual private network” requires direct communication between member computers.[2]

The Court construes “virtual private network” as “a network of computers which privately and directly communicate with each other by encrypting traffic on insecure paths between the computers where the communication is both secure and anonymous.”

virtual private link

VirnetX proposes “a communication link that permits computers to privately communicate with each other by encrypting traffic on insecure communication paths between the computers.” Defendants, except the two Aastra entities, propose “a link in a virtual private network.” The Aastra entities propose “a link in a virtual private network that accomplishes data security and anonymity through the use of hop tables.”

VirnetX’s proposed construction closely tracks its proposal for “virtual private network,” replacing “a network of computers which” with “a communication link that permits computers to.” “Network of computers” implies that the computers are linked together; likewise a “communication link that permits computers [to communicate]” implies a computer network.

Defendants also note the similarity between VirnetX’s proposed construction of “virtual private network” and “virtual private link.” Defendants contend that VirnetX’s proposal is essentially “a communication link that permits computers to VPN.” Tr. of *Markman* Hr’g 55, Jan. 5, 2012. As a simplification, Defendants propose “a link in a virtual private network.”

The Aastra entities argue that a virtual private link should be limited to virtual private network links that use hop tables to achieve data security and anonymity. An embodiment of

[2] Defendants stipulated at the *Markman* hearing that they were not arguing “directly” requires a direct electromechanical connection. *See* Tr. of *Markman* Hr’g 49–50, Jan. 5, 2012. Rather, Defendants maintained that directly requires direct addressability. Thus, routers, firewalls, and similar servers that participate in typical network communication do not impede “direct” communication between a client and target computer.

claim 13 of the '135 Patent, which contains the term "virtual private link," is depicted in Figure 31. A detailed description of this embodiment is also provided in the specification. *See* '135 Patent cols. 44:14–45:35. This description discusses the use of hopping tables; thus, Aastra argues that this limitation should be imported into the claims.

The Court rejects Aastra's attempt to incorporate limitations of a preferred embodiment into the claims. *See Falana v. Kent State Univ.*, 669 F.3d 1349, 1355 (Fed. Cir. 2012) (cautioning against importing limitations from a preferred embodiment into the claims). The specification notes that the use of hopping is one *option* for accomplishing the data security and anonymity features. *See* '135 Patent col. 45:10–13 ("Next, signaling server 3101 issues a request to transport server 3102 to allocate a hopping table (or hopping algorithm *or other regime*) for the purpose of creating a VPN with client 3103" (emphasis added)). Thus, the applicants envisioned alternate methods of implementing data security and anonymity beyond hopping tables, and importing the hopping limitation into the claims is inappropriate.

The patent specification, in the detailed description of Figure 31, uses the term virtual private network and virtual private link interchangeably. *Compare id.* col. 44:37–40 ("When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server . . . ."), *with id.* col. 45:10–13 (noting that the signaling server requests the transport server to create a hopping table for the purpose of "creating a VPN with client 3103."), *and id.* col. 45:32–35 ("After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server 3102 or signaling server 3101."); *see Nystrom v. Trex Co., Inc.*, 424 F.3d 1136, 1143 (Fed. Cir. 2005) ("Different terms or phrases in separate claims may be construed to cover the same subject matter where the written description and prosecution history indicate that such a reading of the terms or phrases is proper."). Finally, VirnetX's and Defendants' proposed constructions of virtual private link are very similar to their proposed constructions for virtual private network. Accordingly, the Court construes "virtual private link" as "a virtual private network as previously defined."

secure communication link

VirnetX proposes "an encrypted communication link." Defendants propose "virtual private network communication link." The parties in *Microsoft* agreed that this term, as used in the '759 Patent, did not require construction because the claims themselves provide a definition of the term. *Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *43. For instance, claim 1 states: "the secure communication link being a virtual private network communication link over the computer network." '759 Patent col. 57:20–22. Here, the parties also agree that, as to the '759 Patent, the term means "virtual private network communication link." However, the claims of the '504 and '211 Patents use this term without further defining it. Thus, the parties dispute the construction of the term as used in the '504 and '211 Patents.

VirnetX contends that "secure" means the link uses some form of data encryption, highlighting the following passage from the '504 Patent specification: "Data security is usually tackled using some form of data encryption." '504 Patent col. 1:55–56. VirnetX argues that the inventors would have used the term "virtual private network communication link" had it desired to limit "secure communication link" to that interpretation. VirnetX further argues Defendants' proposal improperly imports a limitation from the preferred embodiment, which discloses a secure communication link that is also a virtual private network communication link. VirnetX states that "Defendants fail to explain why a secure communication link *must always* be a virtual private network communication link for *all* possible embodiments of the claims." Docket No.

192, at 4. Finally, VirnetX argues that it did not narrow the interpretation of "secure communication link" during the prosecution of the '504 and '211 Patents.

Defendants argue that secure communication link is defined in the Summary of the Invention: "The secure communication link is a virtual private network communication link over the computer network." '504 Patent col. 6:61–62. Defendants further argue that the detailed description of the invention also uses the terms "secure communication link" and "virtual private network communication link" synonymously. Defendants also highlight VirnetX's arguments regarding "secure communication link" while prosecuting U.S. Patent No. 8,051,181 ("the '181 patent"), a related patent that is not at issue in the instant case.

The '181 Patent is related to the patents-in-suit; it is a division of a continuation-in-part of the '783 Application that serves as an ancestor application for all of the patents-in-suit. The Federal Circuit has held that arguments to the PTO regarding one patent application are applicable to related patent applications. *See Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1349 (Fed. Cir. 2004) ("[T]he prosecution history of one patent is relevant to an understanding of the scope of a common term in a second patent stemming from the same parent application."). The Federal Circuit has also held that arguments regarding a later filed application may be applicable to a previously filed application. *See Verizon Servs. Corp. v. Vonage Holdings Corp.*, 503 F.3d 1295, 1307 (Fed. Cir. 2007) (rejecting the argument that a disclaimer should not apply because it occurred after the patent under consideration had issued). Here, the '181 Patent issued after all of the patents-in-suit. Its application was filed later than the applications for the patents-in-suit except for the '211 Patent, which was filed approximately six months earlier.

When prosecuting the '181 Patent, VirnetX distinguished the Aventail reference from the "secure communication link" limitation using arguments nearly identical to those discussed earlier regarding Aventail and the "virtual private network" term. VirnetX argued that Aventail failed to disclose a "secure communication link" for the same three reasons asserted in the '135 reexamination. *Compare* Docket No. 182 Attach. 16, at 5–7 (arguments regarding "virtual private network" and Aventail), *with* Docket No. 202 Attach. 1, at 6–8 (arguments regarding "secure communication link" and Aventail). Therefore, for the same reasons stated earlier regarding "virtual private network," a "secure communication link" also requires direct communication between its nodes.

"Secure communication link" was originally used in the claims of the '759 Patent, which was also at issue in *Microsoft*. There, the parties agreed that it did not require construction because the claim language itself defined the term as "being a virtual private network communication link." '759 Patent col. 57:20–22. However, the later-filed applications that issued as the '504 and '211 Patents removed this defining language from the claims. Accordingly the term is not so limited in the '504 and '211 Patents as in the '759 Patent.

Defendants argue that the Summary of the Invention defined a secure communication link as a virtual private network communication link. However, this discussion in the Summary of the Invention relates to a particular preferred embodiment and opens as follows:

> According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" . . . technique without being required to enter [information] for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link . . . .

'504 Patent col. 6:36–42. Thus, the advantage of being able to seamlessly establish a one-click VPN is provided by "a method for establishing a secure communication link." The description continues by describing the details of an embodiment that realizes this advantage. *See id.* cols. 6:43–7:10 (describing the one-click embodiment). It is within this description of the preferred embodiment that the specification acknowledges that the "secure communication link is a virtual private network communication link." *Id.* col. 6:61–63. The patentee is not acting as his own lexicographer here; rather, he is describing a preferred embodiment. The claims and specification of the '504 and '211 Patents reveal that the patentee made a conscious decision to remove the virtual private network limitation originally present in the '759 Patent claims. Thus, secure communication link shall be interpreted without this limitation in the '504 and '211 Patents.

VirnetX proposes that a secure communication link is an encrypted link. However, claim 28 of the '504 Patent[3] covers "[t]he system of claim 1, wherein the secure communication link uses encryption." '504 Patent col. 57:17–18. VirnetX's proposal seeks to import a limitation from dependent claim 28 into independent claim 1, and this violates the doctrine of claim differentiation. *See Curtiss-Wright Flow Control Corp. v. Velan, Inc.*, 438 F.3d 1374, 1380 (Fed. Cir. 2006) ("'[C]laim differentiation' refers to the presumption that an independent claim should not be construed as requiring a limitation added by a dependent claim."). The specification notes that "[d]ata security is *usually* tackled using some form of data encryption." '504 Patent col. 1:55–56 (emphasis added). Therefore, encryption is not the only means of addressing data security. Accordingly, a secure communication link is one that provides data security, which includes encryption.

The Court construes "secure communication link" as "a direct communication link that provides data security."[4]

[3] Claim 28 of the '211 Patent is similar.

[4] As the Court discussed earlier, the '759 Patent claims further limit the secure communication link recited therein. This construction does not contradict these provisions of the '759 claims, which limit the secure communication link there to a virtual private network communication link. Thus, as a practical matter, the "secure communication link" recited in the '759 Patent claims is a "virtual private network communication link."

domain name service

VirnetX proposes "a lookup service that returns an IP address for a requested domain name," adopting the Court's previous construction of this term in *Microsoft*. Defendants propose to append "to the requester" to VirnetX's proposed construction.

VirnetX argues that Defendants' proposal incorporates an extraneous limitation. Further, VirnetX provides an expert declaration stating that one of skill in the art, after reading the specification, would understand that a domain name service does not necessarily return the requested IP address to the requester. *See* Docket No. 173 Attach. 17 ¶¶ 7–8 (stating that in the context of a DNS proxy, the IP address may be returned to the original requesting client, the proxy, or both). VirnetX also argues that the specification envisions a domain name service that does not always return an address to the requester. For instance, the specification states:

> According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user . . . , the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user.

'135 Patent cols. 37:63–38:2. Defendants argue that VirnetX ignores the implicit meaning of the Court's *Microsoft* construction by arguing that a domain name service does not necessarily return the requested IP address to the requester.

VirnetX's expert explains that "in one mode, the domain name request can be received by a DNS proxy (or DNS proxy module), which, in turn, may forward the request to a DNS function that can return an IP address." Docket No. 173 Attach. 17 ¶ 8. Thus, VirnetX argues, a domain name request may cause an IP address to be returned "to the client, or to a DNS proxy . . . , or both." *Id.* VirnetX's expert is effectively describing a scenario detailed in the '135 Patent and cited above by VirnetX. This scenario is further described in detail in the specification and depicted in Figure 26. *See* '135 Patent col. 38:13–42 (describing the operation of the system depicted in Figure 26). VirnetX asserts that Defendants' proposed construction precludes this preferred embodiment.

Contrary to VirnetX's argument, Defendants' proposed limitation does not preclude a preferred embodiment. The "specialized" or "modified" DNS server referenced in the specification is shown as 2602 in Figure 26. This modified DNS server contains a DNS proxy function and a standard DNS server function. Requests for non-secure sites are passed through to the DNS server, and an IP address is returned to the requesting client. In this case, two separate domain name requests are effectively being made: (1) between the client computer 2601 and the modified DNS server 2602; and (2) between the DNS Proxy 2610 and the DNS Server 2609. If the original client request is for a secure site, then the DNS Proxy 2610 establishes a VPN connection between the client and the secure site. The specification explains the final stages of this process:

> Thereafter, DNS proxy 2610 returns to user computer 2601 the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) 2604, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Id. col. 38:36–42. The DNS Proxy 2610, operating as an internal component of the modified DNS server 2602, returns an address to the requestor, the client computer 2601. Thus, viewing the modified DNS server 2602 as a black box, it returned an address to the requesting client computer.

For these reasons, the Court finds that a domain name service inherently returns the IP address for a requested domain name to the requesting party. The Court construes "domain name service" as "a lookup service that returns an IP address for a requested domain name to the requester."

domain name

VirnetX proposes the same construction adopted by the Court in *Microsoft*: "a name corresponding to an IP address." Defendants propose "a hierarchical sequence of words in decreasing order of specificity that corresponds to a numerical IP address." In *Microsoft*, the Court addressed Defendants' argument that a domain name is necessarily hierarchical in nature; that analysis is incorporated herein. *See Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *24–25. For the same reasons stated in *Microsoft*, the Court construes "domain name" as "a name corresponding to an IP address."

DNS proxy server

VirnetX proposes "a computer or program that responds to a domain name inquiry in place of a DNS." Defendants propose "a computer or program that responds to a domain name inquiry in place of a DNS, and prevents destination servers from determining the identity of the entity sending the domain name inquiry." VirnetX's proposal and the first portion of Defendants' proposal reflect the construction adopted by this Court in *Microsoft*. *Id.* at *39. Here, the dispute is whether a DNS proxy server "prevents destination servers from determining the identity of the entity sending the domain name inquiry."

Defendants derive support for their proposed limitation directly from the Background of the Invention: "Proxy servers prevent destination servers from determining the identities of originating clients." '135 Patent col. 1:49–50. VirnetX argues that this statement should be read in the context of the sentence that precedes it: "To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic." *Id.* col. 1:46–49. VirnetX contends that these statements are not regarding all proxy servers, but merely detail how proxy servers may be configured to achieve anonymity.

VirnetX also argues that adopting Defendants' construction would read out a preferred embodiment disclosed in Figure 26 of the '135 Patent. In Figure 26, user computer 2601, after interfacing with DNS Proxy 2610, communicates directly with Secure Target Website 2604 or Unsecure Target Site 2611. In this configuration, the DNS Proxy does not prevent the destination servers (secure and unsecure target websites) from learning the identity of the originating client (user computer). Rather, the DNS Proxy enables direct communication between the originating client and destination servers. Accordingly, Defendants' proposed additional limitation should be rejected. *See Globetrotter Software, Inc. v. Elan Computer Grp., Inc.*, 362 F.3d 1367, 1381 (Fed. Cir. 2004) ("A claim interpretation that excludes a preferred embodiment from the scope of the claim 'is rarely, if ever, correct.'" (quoting *Vitronics Corp. v. Conceptronic*, 90 F.3d 1576, 1583 (Fed. Cir. 1996))).

For these reasons and those stated in *Microsoft*, *see* 2009 U.S. Dist. LEXIS 65667, at *39–42, the Court construes "DNS proxy server" as "a computer or program that responds to a domain name inquiry in place of a DNS."

secure domain name service

VirnetX proposes "a lookup service that recognizes that a query message is requesting a secure computer address, and returns a secure computer network address for a requested secure domain name." Defendants propose "a non-standard lookup service that recognizes that a query message is requesting a secure computer address, and performs its services accordingly by returning a secure network address for a requested secure domain name." Both parties propose a different construction from that adopted by this Court in *Microsoft* because of arguments made during reexamination of the '180 Patent. The following statements by VirnetX during the reexamination of the '180 Patent provide the basis for both parties' proposals:

> A secure domain name service is not a domain name service that resolves a domain name query that, unbeknownst to the secure domain name service, happens to be associated with a secure name. A secure domain name service of the ‘180 Patent, instead, recognizes that a query message is requesting a secure computer network address and performs its services accordingly.

Docket No. 173 Attach. 13, at 24 (internal citations omitted). The parties dispute whether “non-standard” should characterize the secure domain name service and whether the “perform its services accordingly” language should be included in the construction.

Defendants contend that VirnetX’s reexamination arguments require that a secure domain name service be qualified as a “non-standard” service. During reexamination, VirnetX argued that the PTO’s “position that a secure domain name service is nothing more than a *conventional* DNS server that happens to resolve domain names of secure computers” was faulty. *Id.* (emphasis added). VirnetX clarified that a “secure domain name service is unlike a *conventional* domain name service . . . .” *Id.* (emphasis added). VirnetX further explained that “a secure domain name service can resolve addresses for a secure domain name; whereas, a *conventional* domain name service cannot resolve addresses for a secure domain name.” *Id.* (emphasis added). VirnetX repeatedly distinguishes a secure domain name service from a *conventional* domain name service, implying that the secure domain name service is not conventional. Further, the ‘180 Patent distinguishes between a secure domain name service and a standard domain name service. *See* ‘180 Patent col. 51:29–45 (distinguishing between a “secure domain name service (SDNS)” and a “standard domain name service (STD DNS)”).

VirnetX argues that the “non-standard” limitation is not supported by the specification or prosecution history. However, the ‘180 Patent specification and VirnetX’s statements during the ‘180 reexamination support Defendants’ proposed distinction between a standard domain name service and a secure (non-standard) domain name service. Accordingly, the “non-standard” characterization proposed by Defendants should be retained.

Defendants next argue that "perform its services accordingly" should be included in the construction because this is part of the language that VirnetX used to distinguish a secure domain name service from a conventional domain name service during reexamination. VirnetX responds that this language is superfluous because both parties agree on the task performed by the secure domain name service, namely, returning a secure network address for a requested secure domain name. The Court agrees that "perform its services accordingly" adds little to the understanding of secure domain name service and should not be included in the construction.

The Court construes "secure domain name service" as "a non-standard lookup service that recognizes that a query message is requesting a secure computer address, and returns a secure computer network address for a requested secure domain name."

domain name service system

VirnetX proposes that no construction is necessary, but alternatively proposes "a computer system that includes a domain name service (DNS)." Defendants propose "a DNS that is capable of differentiating between, and responding to, both standard and secure top-level domain names."

Both parties cite claim 1 of the '504 Patent to support their proposed constructions, which states:

> A system for providing a domain name service for establishing a secure communication link, the system comprising:
>
> a *domain name service system* configured to be connected to a communication network, to store a plurality of domain names and corresponding network addresses, to receive a query for a network address, and to comprise an indication that the *domain name service system* supports establishing a secure communication link.

'504 Patent col. 55:49–56 (emphases added). VirnetX argues that the claim language itself describes the required properties of a domain name service system. Defendants argue that the addition of the word "system" to domain name service means it must be something more than a lookup service. Defendants also cite the preferred embodiment disclosed in Figures 33 and 34 of the specification to further support their construction. There, the client computer 3301 is potentially able to communicate with the non-secure server 3304 and the secure server 3320. Accordingly, Defendants argue, the domain name service system, which correlates to the preferred embodiment described in Figures 33 and 34, necessarily includes the ability to handle both secure and standard domain names.

The claims do not require that that domain name service system be able to handle both secure and non-secure domain names. Rather, it must "comprise an indication that [it] supports establishing a secure communication link." *Id.* col. 55:54–56. Defendants seek to improperly import limitations from a preferred embodiment into the claim language. The claim language itself provides a description of the domain name service system. Thus, the Court finds that "domain name service system" does not require construction.

web site

VirnetX proposes "a computer associated with a domain name and that can communicate in a network ." Defendants propose "one or more related web pages at a location on the World Wide Web." These two proposals mirror the proposals made in *Microsoft*. *See Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *26. There, the Court adopted Defendants' proposal.

The parties' arguments mirror those made in *Microsoft*; in fact, VirnetX has incorporated its *Microsoft* arguments by reference. VirnetX notes that the examiner failed to mention web pages or the World Wide Web in determining that the Aventail reference met the "secure web site" limitation of certain claims of the '135 Patent. VirnetX argues that this supports its broader construction of website, showing that a person of ordinary skill in the art would not read web site so narrowly as to implicate the World Wide Web. As both parties have recognized, however, this

Court is not bound by the examiner's evaluation of prior art. The Court hereby incorporates its previous analysis of the arguments regarding the term "web site." *See id.* at *26–31.

For the same reasons stated in *Microsoft*, the Court construes "web site" as "one or more related web pages at a location on the World Wide Web."

secure web site

VirnetX proposes "a computer associated with a domain name and that can communicate in a virtual private network." Defendants propose "a web site that requires authorization for access and that can communicate in a VPN." Again, the proposals and arguments regarding this term mirror those in *Microsoft*, and the Court incorporates by reference its previous analysis. *See id.* at *31–33. The Court construes "secure web site" as "a web site that requires authorization for access and that can communicate in a VPN."

secure target web site

VirnetX proposes "a target computer associated with a domain name and that can communicate in a virtual private network." Defendants propose "a secure web site on the target computer."

Claim 1 of the '135 Patent, in relevant part, states:

> A method of transparently creating a virtual private network (VPN) between a client computer and a *target computer*, comprising the steps of:
> (1) generating from the client computer a Domain Name Service (DNS) request . . . ;
> (2) determining whether the DNS request transmitted in step (1) is requesting access to a *secure web site*; and
> (3) in response to determining that the DNS request in step (2) is requesting access to a *secure target web site*, automatically initiating the VPN between the client computer and *target computer*.

'135 Patent col. 47:20–33 (emphases added). The method can be stated differently as: if there is a DNS request for a secure web site, create a VPN between the client computer and target computer. This VPN is presumably for accessing the secure web site, and the target computer is hosting the secure web site. The claim language itself supports this interpretation because step 3 refers to "secure target web site," thus linking the earlier referenced secure web site to the earlier referenced target computer. Accordingly, the Court construes "secure target web site" as "a secure web site on the target computer."

secure web computer

VirnetX proposes "a computer that requires authorization for access and that can communicate in a virtual private network." Defendants propose that the term is indefinite or, alternatively propose "the target computer that hosts the secure web site."

Defendants initially argue that "secure web site" is indefinite for lack of a proper antecedent basis. Claim 10 of the '135 Patent is representative and states:

> A system that transparently creates a virtual private network (VPN) between a client computer and *a secure target computer*, comprising:
>
> [1] a DNS proxy server that receives a request from the client computer to look up an IP address for a domain name, . . . wherein the DNS proxy server generates a request to create the VPN between the client computer and *the secure target computer* if it is determined that access to *a secure web site* has been requested; and
>
> [2] a gatekeeper computer that allocates resources for the VPN between the client computer and *the secure web computer* in response to the request by the DNS proxy server.

'135 Patent col. 48:3–19 (emphases added). The preamble states that a VPN is created "between a client computer and a secure target computer." *Id.* col. 48:3–5. Element 1 of the system, the DNS proxy server, requests to create a VPN "between the client computer and the secure target computer [when] access to a secure web site has been requested." *Id.* col. 48:11–14. Element 2 of the system, the gatekeeper computer, "allocates resources for the VPN between the client computer and the secure web computer." *Id.* col. 48:16–18. Element 2 contains the only reference to "secure web computer."

VirnetX argues that the terms "secure target computer" and "secure web computer" are used interchangeably; thus, "secure target computer" provides the antecedent basis for "secure web computer." Defendants alternatively argue that the term must derive its antecedent basis from "secure target computer" if the term is not found indefinite. The Court finds that the term is not indefinite and derives its antecedent basis from "secure target computer." Reading the claim as a whole, it is clear that "secure web computer" refers to the "secure target computer" discussed earlier in the claim.

As with "web site" and "secure web site," VirnetX's proposal ignores the presence of "web" in the term and its link to the World Wide Web. Claim 10 details a system that may create a VPN between two computers in response to a request for access to a web site. If the request is for a non-secure website, the "DNS proxy server returns the IP address for the requested domain name." *Id.* col. 48:6–11. If the request is for a secure website, the "DNS proxy server generates a request to create the VPN between the client computer and the secure target computer." *Id.* col. 48:12–15. Further, in the event that a VPN is created, the claimed system includes "a gatekeeper computer that allocates resources for the VPN between the client computer and the secure web computer." *Id.* col. 48:16–19. Thus, the claims establish that the secure web computer is a target computer for the VPN connection and is able to respond to a request for access to a secure web site. Accordingly, the Court construes "secure web computer" as "the target computer that hosts the secure web site."

secure server

VirnetX proposes "a server that requires authorization for access and that can communicate in an encrypted channel." Defendants propose "a server that requires authorization for access and communicates in a VPN." The dispute concerns whether the secure server can communicate in an encrypted channel or communicates in a VPN.

Defendants argue that, in the context of the patents-in-suit, the modifier "secure" means that the modified term (i.e., sever or website) operates or communicates on a VPN. VirnetX contends that "secure" must be read in context and that the claim language itself provides the appropriate context for the term "secure server." Only claims of the '151 Patent contain the term. Claim 1 covers "[a] data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and . . . when the intercepted DNS request corresponds to a secure server, *automatically initiating an encrypted channel between the client and the secure server*." '151 Patent col. 46:55–67 (emphasis added). Claim 2, which depends from claim 1, also recites the following method step: "when the client is authorized to access the secure server, sending a request to the secure server *to establish an encrypted channel between the secure server and the client*." *Id.* col. 47:5–8 (emphasis added). Both claims envision creating an encrypted channel between the secure server and requesting client.

Defendants' proposal that a secure server "communicates in a VPN" is too limiting for two reasons. First, the claims only require the creation of an encrypted channel, not a VPN. Second, requiring that a secure server actively "communicates" is not supported by the claim language. On the other hand, VirnetX's proposal recognizes the *possibility* of communication over an encrypted channel, which is supported by the language of the claims. Accordingly, the Court construes "secure server" as "a server that requires authorization for access and that can communicate in an encrypted channel."

target computer

VirnetX argues that no construction is necessary, but alternatively proposes "a computer with which the client computer seeks to communicate." Defendants propose "the ultimate destination with which the client computer seeks to communicate." The parties essentially agree that a target computer is one "with which the client computer seeks to communicate." The dispute pertains to whether target computer needs further limitation.

In general, the claims cover communications among various entities in determining whether a VPN or link should be established. If it is determined that a VPN or link should be established, the claims cover methods of initiating such VPN or link. The term "target computer" is used within this general context. Claim 1 of the '135 Patent captures the essence of this term: "A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising . . . ." '135 Patent col. 47:20–22. Claim 2, which depends from claim 1, covers the use of a DNS server separate from the client computer. *See id.* col. 47:33–35. Claim 8 envisions the use of a DNS proxy server to pass through requests made for non-secure websites. *See id.* col. 47:60–64. Claim 7, which depends from claim 1, adds the "step of using a gatekeeper computer that allocates VPN resources for communicating between the client computer and the target computer." *Id.* col. 47:56–59.

Defendants argue that "target computer" should be limited to the "ultimate destination" of the client computer's intended communication. Defendants contend that omitting this clarification would permit a claim interpretation where the DNS proxy server or gateway computer (referenced in claims 7 and 8) were target computers. VirnetX argues that "nothing in the claim language precludes a communication from going beyond a target computer." Docket No. 192, at 9. VirnetX further argues that establishing a VPN between a client computer and a target computer on a private network would possibly allow the client to communicate with multiple computers on that private network.

One of skill in the art, reading the claims as a whole, would not confuse target computer with the DNS proxy server or gatekeeper computer, as Defendants suggest. Claim 1 clearly indicates that it covers a method for establishing a VPN between a client computer and target computer. The claims and specification discuss the DNS proxy server and gatekeeper server as facilitating that process. Further, the claims do not limit the client computer's communication once a VPN has been established. Thus, Defendants' proposed "ultimate destination" limitation is inappropriate. The Court finds that "target computer" does not require construction.

between [A] and [B]

VirnetX argues that no construction is necessary, and Defendants propose "extending from [A] to [B]." The term "between [A] and [B]" is used in various claims of the patents-in-suit, where A and B refer to computers or locations. The parties do not dispute that all of the "between" phrases should be construed in the same manner. Claim 1 of the '135 Patent provides a representative example of the phrase's use: "A method of transparently creating a virtual private network (VPN) between [A] a client computer and [B] a target computer, comprising . . . ." '135 Patent col. 47:20–22. The following arguments and analysis are presented in light of this representative claim.

Defendants argue that the between phrase requires the VPN to extend from the client computer to the target computer. VirnetX contends that the VPN must only extend along public communication paths because private communication paths are inherently secure. The specification does not reveal or suggest the embodiment hypothesized by VirnetX. One of ordinary skill in the art, in light of the specification and claims, would understand creating a VPN between a client computer and target computer as creating a VPN that extends from the client computer to the target computer. Accordingly, the Court construes "between [A] and [B]" as "extending from [A] to [B]."

generating from the client computer a Domain Name Service (DNS) request

VirnetX contends that no construction is necessary. Defendants propose "creating and transmitting from the client computer a DNS request." At the *Markman* hearing, the parties agreed to the following construction for this term: "generating and transmitting from the client computer a DNS request."

an indication that the domain name service system supports establishing a secure communication link

VirnetX argues that this term does not require construction. Defendants propose "a visible message or signal that informs the user that the domain name service system supports establishing a secure communication link."

Defendants argue that the "indication" must be visible to the user, noting that the preferred embodiments disclose user-visible indications. *See, e.g.*, '504 Patent Figs. 33 & 34 (containing a "Go Secure" hyperlink). VirnetX argues that limitations present in the preferred embodiments should not be imported into the claims.

The specification of the '504 Patent states:

> Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click).

Id. col. 49:6–12. Thus, the specification envisions alternative methods of activating a secure communication link other than clicking a hyperlink, which is necessarily visible. An audible message could be provided to the user indicating that the system supports establishing a secure communication link, and a simple key press may be used to activate such a secure communication link. Neither the specification nor the claim language provides a basis for limiting "indicating" to a visual indicator. The Court finds that this term is readily understandable and does not require construction.

indicate/indicating . . . whether the domain name service system supports establishing a secure communication link

VirnetX argues that this term does not require construction. Defendants propose "display/displaying a visible message or signal that informs the user whether the domain name service system supports establishing a secure communication link." The issue and arguments regarding this term are identical to those raised for the previous term. For the same reasons stated regarding the previous term, the Court finds that this term does not require construction.

enabling a/the secure communication mode of communication

VirnetX argues that this term does not require construction. Defendants propose "using an input device to select a secure communication mode of communication."

Defendants argue that the specification teaches that a secure communication mode of communication is enabled by using an input device such as a mouse or keyboard. *See* '759 Patent col. 6:44–51 (noting that a secure communication mode of communication is preferably enabled by selecting an icon or entering a command). However, the specification also envisions establishing a secure link "as a default setting at boot-up of the computer (i.e., no click)." *Id.* col. 50:28–29. Thus, Defendants' proposal is overly limiting because it requires a user to activate the secure mode of communication. Further, as noted in *Microsoft*, the Court finds that this term is readily understandable to a jury. *See Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *45. Accordingly, the Court finds that this term does not require construction.

cryptographic information

VirnetX proposes "information that is used to encrypt data or information that is used to decrypt data." Defendants propose "information that is required in order to encode/decode or encrypt to ensure secrecy." Both parties agree that cryptographic information concerns information used to perform encryption or decryption as opposed to the underlying data that is encrypted or decrypted.

VirnetX argues that Defendants' proposal is ambiguous for the following two reasons. First, it seems to require the inclusion of all information necessary for encryption, opening a debate about what information is necessary. Second, the phrase "to ensure secrecy" invites an additional dispute about the strength of the encryption enabled by the cryptographic information. Defendants respond that their construction closely tracks that issued by this Court in *Microsoft*.[5] Defendants assert that the phrase "to ensure secrecy" was used in *Microsoft* to clarify the type of encryption at issue (security as opposed to compression) and should likewise be used here for the same reasons.

Claim 1 of the '759 Patent includes the following step: "enabling a secure communication mode of communication at the first computer without a user entering any cryptographic information *for establishing the secure communication mode of communication*." '759 Patent col. 57:10–16 (emphasis added). The claim language itself establishes the purpose of the cryptographic information; thus, the "to ensure secrecy" clarification is not necessary. Also, the claim covers a method that does not require the user to enter any cryptographic information. Further, the parties acknowledge that there are different types of cryptographic information (i.e., username, password, encryption key, etc.). Whether some or all of that information is required for encryption or decryption is application specific. What is important to the claims of the '759 Patent is whether any such information is required by the user "for establishing the secure

[5] In *Microsoft*, the Court construed the term as "information that is encoded/decoded or encrypted to ensure secrecy." *Microsoft*, 2009 U.S. Dist. LEXIS 65667, at *46.

communication mode of communication." Thus, the arguments about how much cryptographic information must be used are irrelevant.

The Court construes "cryptographic information" as "information that is used to encrypt data or information that is used to decrypt data."

CONCLUSION

For the foregoing reasons, the Court interprets the claim language in this case in the manner set forth above. Further, VirnetX Inc.'s Motion to Compel from Apple a Complete Response to VirnetX's Eighth Common Interrogatory (Docket No. 179) is **DENIED AS MOOT**. For ease of reference, the Court's claim interpretations are set forth in a table in Appendix A.

So ORDERED and SIGNED this 25th day of April, 2012.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

APPENDIX A

Claim Term	Court's Construction
virtual private network	a network of computers which privately and directly communicate with each other by encrypting traffic on insecure paths between the computers where the communication is both secure and anonymous
virtual private link	a virtual private network as previously defined
secure communication link	a direct communication link that provides data security
domain name service	a lookup service that returns an IP address for a requested domain name to the requester
domain name	a name corresponding to an IP address
DNS proxy server	a computer or program that responds to a domain name inquiry in place of a DNS
secure domain name service	a non-standard lookup service that recognizes that a query message is requesting a secure computer address, and returns a secure computer network address for a requested secure domain name
domain name service system	No construction necessary
web site	one or more related web pages at a location on the World Wide Web
secure web site	a web site that requires authorization for access and that can communicate in a VPN
secure target web site	a secure web site on the target computer
secure web computer	the target computer that hosts the secure web site
secure server	a server that requires authorization for access and that can communicate in an encrypted channel
target computer	No construction necessary
between [A] and [B]	extending from [A] to [B]
generating from the client computer a Domain Name Service (DNS) request	generating and transmitting from the client computer a DNS request
an indication that the domain name service system supports establishing a secure communication link	No construction necessary
indicate/indicating . . . whether the domain name service system supports establishing a secure communication link	No construction necessary
enabling a/the secure communication mode of communication	No construction necessary
cryptographic information	information that is used to encrypt data or information that is used to decrypt data

TAB 2

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

VIRNETX INC.,	§	
Plaintiff,	§	
vs.	§	**CASE NO. 6:10-CV-417**
CISCO SYSTEMS, INC. et al.,	§	
Defendants.	§	

ORDER

Before the Court are the following motions:

- Apple's Motion for Summary Judgment of Non-Infringement of the '135 and '151 Patents (Docket No. 442);
- Defendants' Motion to Exclude the Expert Opinions of Mr. Roy Weinstein (Docket No. 445);
- Defendants Cisco and Apple's Motion to Stay Pending Ongoing Reexamination Proceedings (Docket No. 477); and
- Joint Motion to Exceed Limits on Exhibit and Deposition Designations for Trial (Docket No. 550).

Having considered the parties written submissions and oral arguments, the Court **DENIES** Apple's Motion for Summary Judgment of Non-Infringement of the '135 and '151 Patents and **DENIES** Defendants' Motion to Exclude the Expert Opinions of Mr. Roy Weinstein with opinions to follow. Additionally, as stated at the hearing, the Court **DENIES** Defendants Cisco and Apple's Motion to Stay Pending Ongoing Reexamination, and the Court **GRANTS** the Joint Motion to Exceed Limits on Exhibit and Deposition Designations for Trial.

Additionally, the parties are given the following trial times:

Jury Selection	30 minutes per side
Opening Statement	30 minutes per side
Direct/Cross	12 hours per side
Closing Argument	45 minutes per side

These times are inclusive of **all issues**, jury and non-jury. It is the parties' responsibility to budget their time accordingly.

So ORDERED and SIGNED this 23rd day of October, 2012.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

TAB 3

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

	§	
VIRNETX INC.,	§	
	§	
Plaintiff,	§	
	§	
vs.	§	**CASE NO. 6:10-CV-417**
	§	
APPLE INC.,	§	
	§	
Defendant.	§	
	§	

MEMORANDUM OPINION AND ORDER

The following motions are before the Court:

- Apple's Motion for Judgment as a Matter of Law under Rule 50(b) or, in the alternative, for a New Trial or a Remittitur (Docket No. 623);
- VirnetX's Motion for Post-Verdict Damages to the Time of Judgment, Pre-Judgment Interest, and Post-Judgment Interest (Docket No. 620);
- VirnetX's Amended Motion for Post-Verdict Damages (Docket No. 657);
- VirnetX's Motion for a Permanent Injunction (Docket No. 621); and
- VirnetX's Motion for Entry of Judgment on the Jury Verdict, Request for Attorneys' Fees, and Judgment against Apple on Apple's Late-Abandoned Counterclaims and Defenses, including all of Apple's Alleged Prior Art References (Docket No. 625).

For the reasons stated below, Apple's Motion for Judgment as a Matter of Law under Rule 50(b) or, in the alternative, for a New Trial or a Remittitur is **DENIED**. VirnetX's Motion for Post-Verdict Damages to the Time of Judgment, Pre-Judgment interest, and Post-Judgment Interest is **GRANTED IN PART** and **DENIED IN PART**. VirnetX's Amended Motion for Post-Verdict Damages is **GRANTED.** VirnetX's Motion for Permanent Injunction is **DENIED**, and

SEVERS VirnetX's request for an Ongoing Royalty into a separate action. Lastly, VirnetX's Motion for Entry of Judgment on the Jury Verdict, Request for Attorneys' Fees, and Judgment against Apple on Apple's Late-Abandoned Counterclaims and Defenses, including all of Apple's Alleged Prior Art References is **GRANTED IN PART** and **DENIED IN PART**.

BACKGROUND

On August 11, 2010, VirnetX, Inc. ("VirnetX") filed this action against Apple, Inc. ("Apple") alleging that Apple infringed U.S. Patent Nos. 6,502, 135 ("the '135 Patent"), 7,418, 504 ("the '504 Patent), 7,490,151 ("the '151 Patent), and 7,921,211 ("the '211 Patent)(collectively, "the patents-in-suit"). The '135 and '151 Patents generally describe a method of transparently creating a virtual private network ("VPN") between a client computer and a target computer, while the '504 and '211 Patents disclose a secure domain name service.

VirnetX accuses Apple's VPN On Demand and FaceTime features of infringement. Both products feature establishing secure communications, with Apple's FaceTime feature providing a secure communication link for users when video-chatting. Apple's VPN On Demand feature on the other hand is a product that seamlessly creates a VPN when a user requests access to a secure website or server.

A jury trial regarding the instant suit commenced on October 31, 2012. At trial, VirnetX contended that Apple infringed claims 1, 3, 7, 8 of the '135 Patent; claims 1 and 13 of the '151 Patent; claims 1, 2, 5, 16, 21, and 27 of the '504 Patent; and claims 36, 37, 47 and 51 of the '211 Patent. In response, Apple asserted its FaceTime and VPN On Demand features did not infringe the patents-in-suit and that the asserted claims were invalid. Following a five-day trial, the jury returned a verdict that the '135, '151, '211, and '504 Patents were not invalid and Apple infringed the asserted claims. To compensate VirnetX for Apple's infringement, the jury awarded VirnetX $368,160,000 in damages.

APPLE'S MOTION FOR JUDGMENT AS A MATTER OF LAW, OR IN THE ALTERNATIVE, FOR A NEW TRIAL OR A REMITTITUR

Judgment as a Matter of Law, New Trial, and Remittitur Standards

Judgment as a matter of law is only appropriate when "a reasonable jury would not have a legally sufficient evidentiary basis to find for the party on that issue." FED. R. CIV. P. 50(A). "The grant or denial of a motion for judgment as a matter of law is a procedural issue not unique to patent law, reviewed under the law of the regional circuit in which the appeal from the district court would usually lie." *Finisar Corp. v. DirectTV Grp., Inc.*, 523 F.3d 1323, 1332 (Fed. Cir. 2008). The Fifth Circuit "uses the same standard to review the verdict that the district court used in first passing on the motion." *Hiltgen v. Sumrall*, 47 F.3d 695, 699 (5th Cir. 1995). Thus, a jury verdict must be upheld, and judgment as a matter of law may not be granted, unless "there is no legally sufficient evidentiary basis for a reasonable jury to find as the jury did." *Id.* at 700. The jury's verdict must also be supported by "substantial evidence" in support of each element of the claims. *Am. Home Assurance Co. v. United Space Alliance*, 378 F.3d 482, 487 (5th Cir. 2004).

A court reviews all evidence in the record and must draw all reasonable inferences in favor of the nonmoving party; however, a court may not make credibility determinations or weigh the evidence, as those are solely functions of the jury. *See Reeves v. Sanderson Plumbing Prods., Inc.*, 530 U.S. 133, 150–51 (2000). The moving party is entitled to judgment as a matter of law, "only if the evidence points so strongly and so overwhelmingly in favor of the nonmoving party that no reasonable juror could return a contrary verdict." *Int'l Ins. Co. v. RSR Corp.*, 426 F.3d 281, 296 (5th Cir. 2005).

Under Federal Rule of Civil Procedure 59, a new trial may be granted to any party to a jury trial on any or all issues "for any reason for which a new trial has heretofore been granted in an action at law in federal court." "A new trial may be granted, for example, if the district court finds the verdict is against the weight of the evidence, the damages awarded are excessive, the trial was unfair, or prejudicial error was committed in its course." *Smith v. Transworld Drilling Co.*, 773 F.2d 610, 612–13 (5th Cir. 1985).

Remittitur is within the sound discretion of the trial court and is only appropriate when the damages verdict is "clearly excessive." *See Alameda Films S.A. v. Authors Rights Restoration Corp.*, 331 F.3d 472, 482 (5th Cir. 2003).

Judgment as a Matter of Law Regarding Direct Infringement

Apple first contends that VirnetX failed to present substantial evidence that Apple's VPN On Demand and FaceTime features infringe the patents-in-suit. *See* Docket No. 623 at 2–16.

Applicable Law

To prove infringement, the plaintiff must show the presence of every element or its equivalent in the accused device. *Lemelson v. United States*, 752 F.2d 1538, 1551 (Fed. Cir. 1985). Determining infringement is a two-step process: "[f]irst, the claim must be properly construed, to determine the scope and meaning. Second, the claim, as properly construed must be compared to the accused device or process." *Absolute Software, Inc. v. Stealth Signal, Inc.*, 659 F.3d 1121, 1129 (Fed. Cir. 2011) (citing *Carroll Touch, Inc. v. Electro Mech. Sys., Inc.*, 15 F.3d 1573, 1576 (Fed. Cir. 1993)). "A determination of infringement is a question of fact that is reviewed for substantial evidence when tried to a jury." *ACCO Brands, Inc. v. ABA Locks Mfr. Co.*, 501 F.3d 1307, 1311 (Fed. Cir. 2007).

VPN On Demand

The parties' primary dispute at trial was if in fact Apple's VPN On Demand feature "determines whether" a DNS request is requesting access to a secure website or server (the "determining whether" limitation/step). At trial, VirnetX alleged the VPN On Demand feature infringed claims 1, 3, 7, 8 of the ’135 Patent, and claims 1 and 13 of the ’151 Patent. The ’135 Patent discloses a method of transparently creating a virtual private network (“VPN”) between client computer and a target computer, while the ’151 Patent describes creating a secure communication link based on a domain name service (“DNS”) request.

Claim 1 of the ’135 Patent is a representative claim and claims the following:

> A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:
>
> (1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;
> (2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and
> (3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer.

While the Court has not construed the word “determining,” in its claim construction opinion, the Court noted this determining step could be performed by the client computer or by the target computer. *See* Docket No. 266 at 18–20. Additionally, the Court has construed the phrase “secure web site” to mean “a website that requires authorization for access and that can communicate in a VPN.” Docket No. 266 at 21.[1]

Apple now contends that VirnetX failed to present substantial evidence that the VPN On Demand feature meets the “determining whether” limitation. Docket No. 623 at 3. Apple argues that the feature cannot meet this limitation, because the accused feature cannot determine whether the DNS request actually corresponds to a secure website or to a secure server, as Apple’s expert, Dr. Kelly, demonstrated. *Id.*; *see* 11/02/12 a.m. TT at 148:10–21; 154:5–156:18; 167:8–176:25.

[1] The ’151 Patent is directed to creating connections with a secure server. Like “secure website,” the Court construed secure server to mean “a server that requires authorization for access and that can communicate in an encrypted channel.” Docket No. 266 at 24.

Apple explains that the VPN On Demand feature operates by referencing a "configuration file," which is created by the user. *Id.* Prior to using the feature, a user will create a list of websites or servers in the "configuration file," which the user wishes to establish a secure connection with in the future. *Id.* at 3. If a particular website is listed in the "configuration file," when a user later requests that particular website, the VPN On Demand feature will first compare the requested website to those websites listed in "configuration file." If the website is listed in the "configuration file," the feature will then automatically create a VPN. *Id.* However, a user may erroneously enter public websites, i.e. non-secure websites, such as www.ebay.com, in the "configuration file." If a user then requests this public website, the feature will automatically create a VPN, even though the website is not a secure website. *Id.* Therefore according to Apple, the VPN On Demand feature does not meet the claim limitation, because the feature does not actually determine whether a website or server is secure. *Id.* Instead, the feature merely compares the requested website to the "configuration file." Apple further illustrates the feature's inability to perform the claim step, explaining when a user requests a secure website that is not listed in the "configuration file," a VPN will not be created even though the user has requested a secure website. *Id.* at 4.

Apple concedes VirnetX did present testimony and evidence that Apple's accused VPN On Demand feature can be configured to infringe the '131 and '151 Patents. *See* 11/01/12 a.m. TT at 48:18–52:15; 56:5–61:9. However, Apple contends VirnetX's expert Dr. Jones's interpretation of the "determining whether" step is unsupported by the claim language. *Id.* At trial, Dr. Jones stated the feature performs the "determining whether" step when the requested website or server is compared to the configuration file. 11/01/12 a.m. TT at 31:19–32:12; 48:18–52:15; 11/01/12 p.m. TT at 18:1–19:20. However, Apple argues that "the claims require that the actual *security* of the requested web site or server be 'determined,'" which does not occur when the requested website or server is compared to the configuration file. *Id.* at 4–5. Apple argues that to be found infringing, the accused VPN On Demand feature must be able to decide if a DNS request corresponds to a secure server or website, but the accused feature lacks this capability. Rather, the feature only initiates VPN connections with whatever server or website the user has provided. *Id.* at 5. Even Dr. Jones acknowledged that the VPN On Demand feature can initiate VPN connections with unsecure websites. *See* 11/01/12 p.m. TT at 21:8–25:7. Since the feature cannot actually determine a requested website or server's security, then Apple argues no reasonable factfinder could have found Apple or its customers infringed the asserted claims of the '151 or the '135 Patent.

However, Dr. Jones presented a thorough infringement analysis of how Apple's VPN On Demand feature can be configured to infringe the '131 and the '151 Patents based on Apple's internal documents and Apple's source code. Dr. Jones explained that the "determining whether" step can be performed by comparing the requested domain name against a list of domain names, i.e. the "configuration file." 11/01/12 a.m. TT at 48:18–52:15. While Dr. Jones acknowledges the feature may initiate VPN connections with unsecure websites, the feature is not intended to be used in this manner. 11/01/12 a.m TT at 26:8–18. In rejecting Apple's alternative, Dr. Jones stated Apple's interpretation would impermissibly add another limitation to the claims, because the claim language does not require verifying the identity of the secure website or server. 11/01/12 p.m. TT at 33:15–35:3.

There is no requirement in the claims, as Apple suggests, that the security of the requested website or server be verified. Instead, the claims merely require that the feature ascertain whether a VPN is needed. *See* '135 Patent col. 47:27–28. Even Apple agrees that if a

Case: 13-1489 Document: 47 Page: 127 Filed: 10/17/2013

user correctly enters a secure website into the "configuration file," the VPN On Demand feature will create a VPN when that website is accessed. *See* 11/02/12 a.m. TT at 153:22–154:4; 11/02/12 p.m. TT at 105:9–12. Simply because a user may enter unsecure websites or servers into the "configuration file" does not defeat infringement. At best, it could have a de minimis effect on damages. Ultimately, Apple is rearguing the infringement theory it presented at trial, which the jury was free to reject. Here, VirnetX produced substantial evidence that the VPN On Demand feature meets the "determining whether" limitation.

Additionally, Apple contends that VirnetX failed to demonstrate that Apple's VPN On Demand feature meets other claim limitations found in either the '135 or '151 Patent. Docket No. 623 at 6–11. Specifically, Apple argues: (1) VirnetX failed to show that the feature returns an IP address for a requested domain name to the requester; (2) failed to show the feature creates a secure and anonymous communication between computers; (3) failed to demonstrate the feature transmits a DNS request from the client computer; and (4) failed to demonstrate that the feature creates an encrypted or secure channel between the client and the secure server. *Id.* at 6, 8–11.

Regarding the encrypted or secure channel limitation, Apple alleges VirnetX's infringement theory is inherently flawed. Docket No. 623 at 10. VirnetX conceded the VPN On Demand feature did not literally meet the "encrypted channel" limitation, however VirnetX argued the feature infringed under the Doctrine of Equivalents ("DOE"). 11/01/12 a.m. TT at 74:6–76:1. Apple argues VirnetX's doctrine of equivalents argument — a system in which the encrypted channel only extends from the device to the VPN server and not to the target server — is flawed, because the proposed equivalent would effectively vitiate the limitation. *Id.* Apple also asserts that the proposed equivalent fails to meet the function-way-result test, because it fails to provide the security one would receive with end to end encryption, as required by the claims. *Id.* at 11.

In response, VirnetX contends Apple's interpretation of the doctrine of equivalents would require VirnetX to prove actual infringement, even though the doctrine clearly does not require such a showing. Docket No. 632 at 10. Also, VirnetX argues Apple's argument regarding a "secure channel" is misplaced, as VirnetX did produce evidence that the limitation was literally practiced by the feature. See 11/01/12 a.m TT at 82:2–23.

To support a finding of infringement under the DOE, a patentee must either: (1) demonstrate an insubstantial difference between the claimed invention and the accused product or method; or (2) satisfy the function, way, result test. *Aquatex Industries, Inc. v. Techniche Solutions*, 479 F.3d 1320, 1326 (Fed. Cir. 2007) (citing *Graver Tank & Mfg. v. Linde Air Prods. Co.*, 339 U.S. 605, 608, 70 S.Ct. 854, 94 L.Ed. 1097 (1950)). A patentee must provide particularized testimony and linking argument as to the insubstantiality of the differences between the claimed invention and the accused device or process on a limitation-by-limitation basis. *Id.* at 1328 (quoting *Texas Instruments, Inc. v. Cypress Semiconductor Corp.*, 90 F.3d 1558, 1567 (Fed. Cir. 1996)). A patentee should typically provide particularized testimony from a qualified expert describing the claim limitations and establishing that those skilled in the art would recognize the equivalents. *Id.* at 1329. However, the expert is not required to "re-start his testimony at square one when transitioning to a doctrine of equivalents analysis." *Paice LLC v. Toyota Motor Corp.*, 504 F.3d 1293, 1305 (Fed. Cir. 2007). Instead, an expert may explicitly or implicitly incorporate his earlier testimony into the DOE analysis. *Id.*

Here, VirnetX provided evidence that the VPN On Demand feature does create an "encrypted channel" as required by claims 1 and 13 of the '151 Patent. *See* 11/01/12 a.m. TT at

74:23–79:10 (Dr. Jones explaining that the creation of an encrypted channel only between the client and VPN server, and not the client and the secure server, meets the claim limitation under DOE). Additionally, VirnetX produced evidence that the feature literally meets the "secure channel" limitation. 11/01/12 a.m. TT at 82:2–23. Accordingly, VirnetX did present substantial evidence to support the jury verdict that the VPN On Demand feature meets the "encrypted channel" and "secure channel" claim limitations.

As to Apple's other arguments, VirnetX did produce substantial evidence showing the Apple's VPN On Demand feature meets the claim limitations disputed by Apple. For example, Dr. Jones explained that the feature returns an IP address for a requested domain name to the requester, when it returns an IP address to the Safari application after a domain was requested. *See* 11/01/12 a.m. TT at 63:13–64:4. He also detailed how a secure and anonymous communication between computers is established by the feature. 11/01/12 a.m. TT at 34:17–37:12; 54:18–55:23. Additionally, Dr. Jones demonstrated how the feature meets the transmitting requirement when the DNS request is sent from the Safari web browser and sent to the iOS operating system. 11/01/12 a.m. TT at 47:22–48:12. Accordingly, VirnetX produced substantial evidence to demonstrate Apple's VPN On Demand meets all the claim limitations of the asserted claims of the '135 Patent and '151 Patent.

Finally, Apple alleges VirnetX failed to demonstrate that Apple directly infringes any asserted method claim of the '135 Patent.[2] Docket No. 623 at 6–7. Apple argues VirnetX

[2] Independent claim 1 and dependent claims 3, 7, and 8 of the '135 Patent were asserted against Apple. Claim 1 is directed to "[a] method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of: (1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer; (2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and (3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer."

presented no evidence that Apple itself preformed all the steps of any asserted claim, and it is improper for VirnetX to assume Apple conducted internal testing of the method while the feature was being developed to show direct infringement. *Id.* at 7; *see Mirror World, LLC v. Apple, Inc.*, 784 F. Supp. 2d. 703, 713 (E.D. Tex. 2011).

"To infringe a method claim, a person must have practiced all steps of the claimed method." *Finjan, Inc. v. Secure Computing Corp.*, 626 F.3d 1197, 1206 (Fed. Cir. 2010). "[A] method claim is not directly infringed by the sale of an apparatus even though it is capable of performing only the patented method. The sale of the apparatus is not a sale of the method. A method claim is directly infringed only by one practicing the patented method." *Joy Techs., Inc. v. Flakt, Inc.*, 6 F.3d 770, 774–75 (Fed. Cir. 1993); *Ricoh Co. v. Quanta Computer Inc.*, 550 F.3d 1325, 1335 (Fed. Cir. 2008) ("[A] party that sells or offers to sell software containing instructions to perform a patented method does not infringe the patent under § 271(a).").

Here, VirnetX did address direct infringement of the '135 Patent, stating Apple performed its own internal testing and use. 11/01/12 a.m. TT at 88:12–89:11 (Dr. Jones stated, "they directly infringe by performing – Apple does that by performing the steps itself, say, through its own internal use and testing."). VirnetX also introduced evidence that once the feature is set-up by the user, the actual steps of the asserted claims are performed by the feature itself and not the user. *See id.;* 11/01/12 a.m. TT at 34:5–20. While Apple contends user input is required to practice the claimed invention, the steps of the asserted claims do not explicitly require an additional party's involvement. *See SiRF Tech., Inc. v. ITC*, 601 F.3d 1319, 1331 (Fed. Cir. 2010)(finding direct infringement when the accused products automatically perform all the steps in the asserted claims). For example, claim 1 of the '135 Patent only has three steps: generating a DNS request; determining whether that DNS request is requesting access to a secure website; and automatically initiating a VPN if the DNS request corresponds to a secure website. Col. 47:21–33. Nothing in the claim language requires a user to set up a configuration file or take other steps for the VPN On Demand feature to infringe. Instead, once a website is entered in a browser, the accused feature will perform the steps of the asserted claims. In light of Dr. Jones' testimony and the claim language, VirnetX provided substantial evidence to establish Apple directly infringed the method claims of the '135 Patent.

FaceTime

As to the FaceTime feature, the parties' arguments at trial mostly concerned whether the feature meets the "direct communication" limitation. At trial, VirnetX alleged Apple's FaceTime feature infringed claims 1, 2, 5, 16, 21, 27 of the '504 Patent and claims 36, 37, 47, and 51 of the '211 Patent. Each claim requires some indication that "the domain name service system supports establishing a secure communication link." *See* '211 Patent, claim 36; '504 Patent, claim 1. In its claim construction opinion, the Court construed "secure communication link" to mean "a direct communication link that provides data security." Docket No. 266 at 13.

Both parties agree that "secure communication" requires a direct communication, however the parties dispute whether a network address translator (NAT) allows for direct communication. *See* 11/01/12 p.m. TT at 50:19–51:15. Both sides acknowledge that the FaceTime feature can operate either via a relay server or via a NAT or other devices containing a NAT functionality. *See* 11/01/12 p.m. TT at 51:24–52:24. VirnetX concedes that the feature does not infringe if calls are routed through a relay server, because there is no direct communication through a relay server. 11/01/12 p.m. TT at 52:19–24. The only dispute is whether a NAT impedes direct communication between the devices.

Apple devices, such as iPhones, can be located behind other devices that have a NAT functionality. NATS alter or readdress the IP address of data packets sent between devices.

Apple argues this readdressing of data packets interrupts any direct communication between the two Apple devices. Docket No. 623 at 11–13.

VirnetX's expert, Dr. Jones, agreed that "secure communication link" requires a direct communication. 11/01/12 p.m. TT at 50:19–22. However, Dr. Jones explained in his testimony that NATs do not impede direct communication. 11/01/12 a.m. TT at 116:22–117:5; 119:8–23. Dr. Jones distinguished a relay server, which VirnetX conceded did not infringe, from a NAT router, which VirnetX argued does infringe. Dr. Jones explained that the relay server creates two separate communications, while a NAT router still allows for "end-to-end communication between the two devices" because it merely translates addresses. 11/01/12 a.m. TT at 116:1–117:5. VirnetX also submitted evidence that describes how the FaceTime feature establishes peer-to-peer connections. *See* PX 213; PX 215; PX 472; PX 485; and PX 478; 11/01/12 a.m. TT at 119:24–123:1.

Additionally, the Court noted in its claim construction opinion that "routers, firewalls, and similar servers that participate in typical network communication do not impede 'direct' communication between a client and a target computer." Docket No. 266 at 8 (addressing the term "directly" in the context of "virtual private network"). VirnetX produced evidence that a NAT operated like a router, firewall or other similar servers and did not impede direct communication. 11/01/12 p.m. TT at 60:12–61:17; 120:13–123:10. Here, the jury was free to accept this testimony or reject some or all of Dr. Alexander's testimony. In light of the foregoing, there was substantial evidence to support a finding that NATs allow direct communication.

Apple also alleges that VirnetX failed to produce sufficient evidence that the feature satisfies the "look up service" limitation[3], that the feature indicates that it supports establishing a secure communication link[4], and that the feature meets the domain name limitation[5], as required by the '504 and '211 Patents. Docket No. 623 at 13–16. However, VirnetX provided a detailed analysis how the FaceTime feature meets every single claim limitation. VirnetX presented evidence that the feature meets the "lookup service" limitation when Apple's FaceTime servers return an IP address for the requested domain name, i.e. phone number or e-mail address. *See* 11/01/12 a.m. TT at 110:20-113:12; PX 249; PX 548. Additionally, VirnetX introduced evidence that the feature provides an indication, when FaceTime provides an accept push message which contains the information necessary to establish a secure communication link. 11/01/12 a.m. TT at 107:15–108:12; 122:9–23; PX 215. Dr. Jones also explained how email addresses or telephone numbers are domain names as required by claims, since e-mail address and telephone numbers have corresponding IP addresses. 10/31/12 p.m. TT at 197:3–20; 11/01/12 a.m. TT at 114:9–115:1.

Conclusion

In light of the foregoing, Apple's motion for judgment as a matter of law regarding direct infringement is **DENIED.**

[3] The claim term "domain name service" was construed to mean "a lookup service that returns an IP address for a requested domain name to the requester." Docket No. 266 at 15. However, the Court did not construe "domain name service system" because the claim language itself provided a description of the term, i.e. that it must "comprise an indication that [it] supports establishing a secure communication link." *Id.* at 20.

[4] Claim 36 of the '211 Patent for example claims a "A non-transitory machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for: connecting the domain name service system to a communication network; storing a plurality of domain names and corresponding network addresses; receiving a query for a network address; and **indicating in response to the query whether the domain name service system supports establishing a secure communication link**."(emphasis added).

[5] The claim term "domain name" was construed to mean a "name corresponding to an IP address." Docket No. 266 at 16.

Judgment as a Matter of Law Regarding Indirect Infringement

Apple also moves for judgment as a matter of law with regard to VirnetX's claims of indirect infringement, arguing VirnetX failed to cite sufficient evidence to support a finding that Apple induced its customers to infringe the patents-in-suit. Docket No. 623 at 17.

Applicable Law

Infringement is a question of fact that is reviewed for substantial evidence when tried to a jury. *Finisar Corp. v. DirecTV Group, Inc.*, 523 F.3d 1323, 1332 (Fed. Cir. 2008). Under 35 U.S.C § 271 (b), a party is liable for infringement if it "actively induces infringement of a patent." "In order to prevail on an inducement claim, the patentee must establish first that there has been direct infringement, and second that the alleged infringer knowingly induced infringement and possessed specific intent to encourage another's infringement." *ACCO Brands, Inc. v. ABA Locks Mfr. Co.*, 501 F.3d 1307, 1312 (Fed. Cir. 2007) (internal quotation marks omitted). Thus, to support a finding of inducement there must be "evidence of culpable conduct, directed to encouraging another's infringement, not merely that the inducer had knowledge of the direct infringer's activities." *DSU Med.Corp. v. JMS Co.*, 471 F.3d 1293, 1306 (Fed. Cir. 2006). Furthermore, "[t]he plaintiff has the burden of showing that the alleged infringer's actions induced infringing acts," and that the infringer "knowingly induced infringement." *Id.* at 1306. A party has the requisite knowledge if it knew "that the induced acts constitute patent infringement," or was willfully blind to the infringement. A party is willfully blind if it believed there was a high probably that the acts constituted patent infringement and took deliberate steps to avoid learning of the infringement. *Global-Tech Appliances, Inc. v. SEB S.A.*, --- U.S. ----, 131 S. Ct. 2060, 2068, 2070 (2011); *see Smith & Nephew, Inc. v. Arthrex, Inc.*, 2013 U.S. App. LEXIS 1038, *10–12 (Fed. Cir. 2013).

Importantly however, a patentee may prove both indirect infringement and the corresponding direct infringement by circumstantial evidence. *See Liquid Dynamics Corp. v. Vaughan Co.*, 449 F.3d 1209, 1219 (Fed. Cir. 2006). “There is no requirement that direct evidence be introduced, nor is a jury's preference for circumstantial evidence over direct evidence unreasonable per se.” *Id.* Moreover, “[t]he drawing of inferences, particularly in respect of an intent-implicating question . . . is peculiarly within the province of the fact finder that observed the witnesses.” *Rolls-Royce Ltd. v. GTE Valeron Corp.*, 800 F.2d 1101, 1110 (Fed. Cir. 1986); *see also Fuji Photo Film Co. v. Jazz Photo Corp.*, 394 F.3d 1368, 1378 (Fed. Cir. 2005) (declining to disturb jury's verdict because intent to induce infringement “is a factual determination particularly within the province of the trier of fact”).

Regarding inducement, Apple argues VirnetX failed to demonstrate that Apple was willfully blind to its customers’ infringement, that Apple was aware of the patents prior to lawsuit, and that Apple intentionally encouraged its customers to infringe the patents-in-suit. Docket No. 623 at 17–20. Apple concedes that VirnetX did present evidence that it was willfully blind to its customers’ infringement, but it contends this evidence fails to support any of VirnetX’s contentions. Essentially, Apple is asking the Court to reweigh the evidence that was before the jury.

VirnetX did produce substantial evidence that Apple induced its customers’ infringement of the patents-in-suit. It presented testimony that Apple believed there was a high probability that its customers infringed and took deliberate actions to avoid confirming infringement. 11/01/12 a.m. TT at 37:14–44:23; 89:21–96:6; 96:18–97:5; 101:13–104:22; 138:17–140:3; 11/02/12 a.m. TT at 14:2–16:6; 17:1–20:9; 25:13–27:3; 30:1–32:1; 33:2–34:24; 50:14–54:10. Apple argues this evidence merely shows that Apple understood how its product worked, had notice of the patents after the suit was filed, and that some of its employees avoided reviewing the patents-in-suit. However, a jury is allowed to draw inferences from the evidence presented. Here, VirnetX presented evidence that Apple believed there was a high probability that the technology at issue was patented. *See Global-Tech*, 131 S. Ct. at 2072; 11/01/12 p.m. TT at 114:1–119:20. Also, VirnetX presented evidence of a clear pattern of behavior by Apple employees, which the jury was free to consider when determining whether Apple took deliberate steps to avoid confirming infringement. Apple argues this is insufficient, because VirnetX failed to present testimony from individuals who routinely review patents; however, a jury is allowed to infer Apple possessed the requisite intent from all the facts as presented. *Broadcom Corp. v. Qualcomm, Inc.*, 543 F.3d 683 (Fed. Cir. 2008)(citing *Water Techs. Corp. v. Calco, Ltd.*, 850 F.2d 660, 669 (Fed. Cir. 1988)("The requisite intent to induce infringement may be inferred from all of the circumstances."). Lastly, as Apple acknowledges, VirnetX presented evidence on how Apple instructs its customers to use the allegedly infringing features. *See* 11/01/12 a.m. TT at 93:20–94:18; 138:5–16. The fact that Apple instructed its customers to use the accused features in an infringing manner adequately supports the jury's finding. *See Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1318–19 (Fed. Cir. 2009)(upholding a jury verdict of induced infringement where the accused inducer designed the accused products to be used in an infringing manner and instructed its customers to use the accused products in an infringing manner).

Accordingly, Apple's Judgment as a Matter of Law as to Indirect Infringement is **DENIED.**

Judgment as a Matter of Law as to Invalidity

Apple argues that no reasonable jury could have found that Takahiro Kiuchi's 1996 publication (the "Kiuchi reference") does not anticipate the asserted claims of the patents-in-suit.

Docket No. 623 at 21. Specifically, Apple contends that its expert, Dr. Alexander, identified each and every claim limitation of the asserted claims in the Kiuchi reference, such that no reasonable jury could have found the patents-in-suit to be valid.

Applicable Law

Patents are presumed valid and overcoming this presumption requires clear and convincing evidence. *Ariad Pharm., Inc. v. Eli Lilly & Co.*, 598 F.3d 1336, 1354 (Fed. Cir. 2010) (*en banc*). A patent claim is invalid as anticipated if the claimed invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention by the applicant. 35 U.S.C. § 102(a) (2006). Anticipation requires the presence in the prior art of each and every limitation of the claimed invention. *Amgen, Inc. v. Hoffman-La Roche Ltd.*, 580 F.3d 1340, 1366 (Fed. Cir. 2009).

Analysis

Upon review of the record, there is substantial evidence to support the jury's verdict regarding anticipation. VirnetX presented evidence and testimony that the Kiuchi reference failed to disclose several claim limitations including, a "secure communication link," "virtual private network," "a DNS proxy server," "an encrypted channel," and "secure channel." Docket No. 632 at 20–21. Also, the jury may have simply disregarded some or all of Dr. Alexander's testimony, when finding the patents-in-suit were not anticipated.

The main dispute between the parties is whether the Kiuchi reference discloses a "secure communication link," i.e. a direct communication. Dr. Alexander opined that the Kiuchi reference does disclose this limitation, because there is direct communication between the client-side proxy and the origin server. *See* 11/02/12 p.m. TT at 167:22–168:12; 176:10–177:20. However, VirnetX contested this interpretation and presented evidence and testimony to the contrary. Dr. Jones, VirnetX's expert, conceded that while the connection between the client-side proxy and the origin server may be direct, the relevant connection is in fact the connection between the client computer and the origin server. *See* 11/05/12 p.m. TT at 44:24–45:22. Dr. Jones explained that this connection —the one between the client computer and the origin server — is not direct because the two devices are separated by two HTTP proxy servers. These proxy servers stop and process the communications between the devices before initiating new connections to forward on, thus interrupting any direct connection between the two devices. Docket No. 632 at 20; *see* 11/05/12 a.m. TT at 92:10–93:4.

Apple contends this distinction is irrelevant, because Dr. Alexander explained that both the user and the client-side proxy could be client computers, as required by the patents-in-suit. Docket No. 623 at 23 (citing 11/05/12 a.m. TT at 47:15–25, 51:18–52:26). However, there was evidence presented that one skilled in the art would not have identified the client-side proxy as a client computer. 11/05/12 a.m. TT at 45:19–50:24; 11/05/12 p.m. at 44:7–44:15. When asked specifically about the connection between the client computer and the origin server, Dr. Alexander failed to explain whether this was truly a direct connection. *See* 11/05/12 a.m. TT at 65:4–8. Here, the jury was presented with conflicting evidence as to whether the Kiuchi reference disclosed a direct connection, as required by the asserted claims. The jury was free to resolve the dispute in VirnetX's favor or disbelieve Dr. Alexander's testimony regarding the issue.

Additionally, the parties disagree whether the Kiuchi reference actually discloses "a plurality of domain names and corresponding network addresses." Docket No. 623 at 22; Docket No. 641 at 6. At trial, Dr. Jones explained that even though the Kiuchi reference disclosed this limitation, the disclosure should not be considered. *See* 11/05/12 p.m. TT at 39:11–42:1; 81:10–

86:9. Dr. Jones contended that the Kiuchi reference was not enabling, because the system described would not actually work. Dr. Jones stated this error was apparent when one examined the appendix of the Kiuchi reference. *Id.* Apple argues it was improper for VirnetX's expert, Dr. Jones, to consider extrinsic evidence, when opining that the Kiuchi reference was not an enabling reference. Docket No. 623 at 22.

To anticipate, a prior art reference not only must disclose each and every limitation of the claimed invention, it must also "enable one of ordinary skill in the art to make the invention without undue experimentation." *In re Gleave*, 560 F.3d 1331 at 1334 (Fed. Cir. 2009). "[I]f the allegedly anticipatory disclosures cited as prior art are not enabled," the claimed invention is not anticipated. *Elan Pharms., Inc. v. Mayo Founadation for Medical Ed. And Research*, 346 F.3d 1051 at 1054 (Fed. Cir. 2003). "In order to show that it is entitled to judgment as a matter of law on its affirmative defense of invalidity, Apple must prove the essential elements of that defense to a virtual certainty." *Mirror Worlds, LLC v. Apple, Inc.*, 784 F. Supp. 2d 703, 729 (E.D. Tex. 2011) (citing *ClearValue, Inc. v. Pearl River Polymers, Inc.*, 735 F. Supp. 2d 560, 577 (E.D. Tex. 2010; *Bank of La. v. Aetna U.S. Healthcare Inc.*, 468 F.3d 237, 241 (5th Cir. 2006)). Here, there is a clear dispute about whether or not the Kiuchi reference is enabling even when the extrinsic evidence is not considered.

For each of the reasons stated above, Apple has not demonstrated it is entitled to judgment as a matter of law as to invalidity.[6]

Judgment as a Matter of Law Regarding Damages

Apple contends that the Court should grant judgment as a matter of law regarding damages, or in the alternative, order a new trial or a remittitur, because the jury's damages award

[6] Apple did not argue at trial that the patents-in-suit were obvious in light of the prior art. Accordingly, this opinion only addresses anticipation.

has no legally sufficient basis and is unsupported by substantial evidence. Docket No. 623 at 25–37. Apple also alleges VirnetX is not entitled to damages prior to when Apple received actual notice of the asserted patents. *Id.* at 37–39.

<u>*Applicable Law*</u>

A patentee is entitled to damages for infringement under 35 U.S.C. § 284. ("Upon finding for the claimant the court shall award the claimant damages adequate to compensate for the infringement, but in no event less than a reasonable royalty for the use made of the invention by the infringer, together with interest and costs as fixed by the court." *Id.*). The burden of proving damages falls on the patentee. *Dow Chem. Co. v. Mee Indus., Inc.*, 341 F.3d 1370, 1381 (Fed. Cir. 2003). There are two alternative categories of infringement compensation: the patentee's lost profits, and the reasonable royalty the patentee would have received through arms-length bargaining. *Lucent Tech., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1324 (Fed. Cir. 2009).

To ascertain the reasonable royalty, patentees commonly consider a hypothetical negotiation, in which the asserted patent claims are assumed valid, enforceable, and infringed, and attempt to ascertain the royalty upon which the parties would have agreed had they successfully negotiated an agreement just before infringement began. *Id.* at 1324–25; *see also Rite-Hite Corp. v. Kelley Co.*, 56 F.3d 1538, 1554 n.13 (Fed. Cir. 1995) (en banc). Calculation of a reasonable royalty requires determination of two separate and distinct amounts: 1) the royalty base, or the revenue pool implicated by the infringement; and 2) the royalty rate, or the percentage of that pool "adequate to compensate" the plaintiff for the infringement. *Cornell Univ. v. Hewlett-Packard Co.*, 609 F. Supp. 2d 279, 286 (N.D.N.Y. 2009).

The entire market value rule "recognizes that the economic value of a patent may be greater than the value of the sales of the patented part alone." *See King Instruments Corp. v.*

Perego, 65 F.3d 941, 951 n.5 (Fed. Cir. 1995). "The entire market value rule allows a patentee to assess damages based on the entire market value of the accused product [if] the patented feature creates the 'basis for customer demand' or 'substantially create[s] the value of the component parts.'" *Uniloc USA, Inc. v. Microsoft Corp.*, 632 F.3d 1292, 1318 (Fed. Cir. 2011) (citing *Lucent*, 580 F.3d at 1336). "[T]he patentee . . . must in every case give evidence tending to separate or apportion the defendant's profits and the patentee's damages between the patented feature and the unpatented features, and such evidence must be reliable and tangible, and not conjectural or speculative," or show that "the entire value of the whole machine, as a marketable article, is properly and legally attributable to the patented feature." *Id.* (citing *Garreston v. Clark*, 111 U.S. 120, 121, 4 S.Ct. 291, 28 L.Ed. 371 (1884)); *see also Lucent*, 580 F.3d at 1336–67. For minor patent improvements, a patentee cannot justify using the entire market value of an accused product simply by asserting a lower royalty rate. *Uniloc*, 632 F.3d at 1319–20 (rejecting contrary interpretation of *Lucent*, 580 F.3d at 1338–39). Although a reasonable royalty analysis "necessarily involves an element of approximation and uncertainty," *Unisplay, S.A. v. Am. Elec. Sign Co.*, 69 F.3d 512, 517 (Fed. Cir. 1995), the Court must ensure that the jury verdict is supported by sufficient evidence.

"A district court's duty to remit excessive damages is a procedural issue, not unique to patent law." *Imonex Servs., Inc. v. W.H. Munzprufer Dietmar Trenner GMBH*, 408 F.3d 1374, 1380 (Fed. Cir. 2005). In the Fifth Circuit, a decision on remittitur and new trial is within the sound discretion of the trial court. *See Volger v. Blackmore*, 352 F.3d 150, 154 (5th Cir. 2003). The standard is highly deferential, and damages are set aside "only upon a clear showing of excessiveness." *i4i Ltd. v. Microsoft Corp.*, 598 F.3d 831, 857 (Fed. Cir. 2010) (quoting *Duff v. Werner Enters., Inc.*, 489 F.3d 727, 730 (5th Cir. 2007)). An excessive award exceeds the "maximum amount calculable from the evidence." *Carlton v. H.C. Price Co.*, 640 F.2d 573, 579 (5th Cir. 1981).

Lack of Evidence as to Direct Infringement

Apple contends there is no legally sufficient basis to award damages for direct infringement as to the '504, '211, and '135 Patents, because VirnetX failed to demonstrate Apple directly infringed the '504, '211, and '135 Patents. Docket No. 623 at 25. Apple's argument is premised on the Court granting Apple judgment as a matter law as to indirect infringement. *Id.* at 25–27. Because the Court has already denied this request, Apple's argument regarding any lack of evidence is unpersuasive.

Lack of Substantial Evidence to Support the Jury's Damages Award

Apple also argues it is entitled to judgment as a matter of law with respect to damages, because VirnetX's damages theory invoked the entire market value rule without complying with the rule, VirnetX's reasonable royalty rate of 1% was not supported by the evidence, and VirnetX's alternative damages model, the Nash Bargaining Solution, does not support the damages award. Docket No. 623 at 27–39.

As an initial matter, the jury departed from both damages models presented at trial. VirnetX's damages model set a floor of $708 million, while Apple proposed a lump sum award of $9.1 million. 11/01/12 p.m. TT at 92:24–93:1; 11/05/12 a.m. TT at 155:19–25. After consideration of the evidence presented at trial, the jury awarded a lump sum of $368,160,000 in damages.

1. Entire Market Value Rule

Apple argues that VirnetX invoked the entire market value rule without actually complying with the rule, because Mr. Weinstein, VirnetX's damages expert, considered the entire value of Apple's devices without demonstrating that the accused features created the basis for customer demand or substantially created the value of the component parts. Docket No. 623 at 27. Apple cites to Mr. Weinstein's testimony where he acknowledged that the allegedly infringing features merely contribute to the sale of the accused devices. *See* 11/01/12 p.m. TT at 170:21–171:21; 222:10–13. Apple contends that even if the infringing features did in fact drive demand for the accused devices, it is irrelevant because the patents-in-suit only cover a small portion of the infringing features, therefore the entire market value rule still cannot be considered. *See* 10/31/12 p.m. TT at 17:5–18:12; 11/01/12 p.m. TT at 166:21–25.

In response, VirnetX argues Apple is misconstruing its damages theory, as VirnetX's primary damages theory sought to determine a reasonable royalty, considering the revenues from the smallest saleable unit of the accused products. Docket No. 632 at 23; s*ee* 11/01/12 p.m. TT at 124:13–20. VirnetX never disclosed the entire revenue of Apple's accused devices, but rather used an apportioned base, which excluded roughly 20% of the total revenue from the accused products, when calculating a reasonable royalty. *See* 11/01/12 p.m. TT at 125:1–22; 128:14–129:8. VirnetX contends it was permissible for its expert to consider the smallest saleable infringing unit when determining the royalty base since the unit had a "close relation to the claimed invention." Docket No. 623 at 25 (quoting *Cornell Univ. v. Hewlett-Packard Co.*, 609 F. Supp. 2d 279, 288 (N.D. NY 2009); *see LaserDynamics, Inc. v. Quanta Computer, Inc.*, 694 F.3d 51, 67 (Fed. Cir. 2012).

Apple counters that if VirnetX wished to calculate its royalty based on the smallest saleable patent-practicing unit, VirnetX needed to demonstrate a close relation between the accused devices and the patented inventions, which it failed to do here. Docket No. 636 at 11; *see* 10/31/12 p.m. TT at 17:5–19:8. However, VirnetX did provide some evidence that the infringing features practicing the patented inventions necessarily utilized other aspects of the accused devices. *See* 10/31/12 a.m. TT at 66:1–22; 89:9–93:6; 100:3–101:22; 106:19–107:2; 11/01/12 a.m. TT at 47:22–55:18.

Additionally while Apple contests the royalty base, it failed to advance a credible alternative. At trial, Apple argued the royalties should be calculated using $29 as the base for all the accused devices. This price was the price of a software upgrade for the Mac computers. *See* 11/05/12 a.m. TT at 147:16–149:6 (Mr. Vellturo, Apple's damages expert, noting that this upgrade allowed the computers to use the accused features).[7] However, Apple never charged its customers for a similar upgrade for its iOS devices, including the iPhone. 11/05/12 p.m. TT at 28:20–29:7. Apple never produced any other testimony or evidence regarding how one should properly calculate the damages for the iOS devices, instead it relied solely on the upgrade price for a dissimilar product. If Apple had advanced a different theory showing the iOS device was not the smallest saleable unit, there may have been a different result.

In light of the foregoing, VirnetX did not implicitly invoke the entire market value rule in its damages theory.

2. Reasonable Royalty Rate

Apple also contends Mr. Weinstein's proposed 1% royalty rate is unsupported by the evidence. Docket No. 623 at 33. To support his 1% royalty rate, Mr. Weinstein considered several licenses, including the In-Q-Tel license, the SafeNet/SAIC agreement, the SAIC/VirnetX agreement, the Microsoft license, the Aastra license, the Mitel license, and the NEC license. Apple argues that almost all of these licenses should not be considered as they postdate the hypothetical negotiation of June 2009. *Id.* 11/01/12 p.m. TT at 201:23–202:9. Without these

[7] VirnetX's damages expert, Mr. Weinstein, did use $29 as his base when calculating damages related to the Mac computers. *See* 11/05/12 a.m. TT at 148:5–13.

licenses, Apple contends there were no comparable licenses to consider at the time of the negotiation. *Id.* For example, the In-Q-Tel license predates even the conception of the patents, while the SafeNet/SAIC agreement was canceled prior to any payment from SafeNet, and the SAIC/VirnetX license was more than a mere license, as it transferred interest of the patents-in-suit. Docket No. 623 at 33; 11/01/12 p.m. TT at 186:9–187:12, 202:5–205:12.

Additionally, Apple criticizes Mr. Weinstein's reliance on VirnetX's licensing policy, which it contends at the time of the hypothetical negotiation was merely a "wish" rather than an actual policy. *See* 11/01/12 p.m. TT at 160:22–161:9. Apple contends there is no evidence that it would have agreed to pay a running royalty in line with this policy, especially in light of the Microsoft agreement, which VirnetX admits did not conform to its licensing policy. 10/31/12 p.m. TT at 137:8–21. Apple also argues Mr. Weinstein did not properly consider Apple's potential non-infringing alternatives. Docket No. 623 at 34. At trial, VirnetX conceded that Apple's FaceTime feature did not infringe the patents-in-suit if the calls were made via a relay server. 11/01/12 a.m. TT at 98:5–14; 99:3–100:1. Even Mr. Weinstein acknowledged the importance of considering non-infringing alternatives in the reasonable royalty analysis. 11/01/12 p.m. TT at 209:2–11. However, Apple argues neither VirnetX nor Mr. Weinstein attempted to assess the feasibility of these alternatives before dismissing their relevancy. Docket No. 623 at 35.

VirnetX did present substantial evidence at trial to support its proposed royalty rate. Mr. Weinstein provided a detailed reasonable royalty analysis to the jury. 11/01/12 a.m. TT at 97:23–122:8. VirnetX presented evidence why such a royalty rate would be justified in light of the value of the technology, noting the accused features were key to Apple attracting potential enterprise customers that valued security in their communications and offered the ease of operation valued by Apple customers. 11/01/12 p.m. TT at 114:1–119:20. Indeed, the accused features have been used extensively by Apple customers, with over 2.6 billion FaceTime calls being made since June 2010. 11/01/12 p.m. TT at 119:13–20. Additionally, VirnetX introduced several license agreements and its own licensing policy to further support the royalty rate. While some of these license agreements postdate the hypothetical negotiation, they help demonstrate what entities would be willing to pay for the technology at issue. *See* 11/01/12 p.m. TT at 108:7–110:16. VirnetX also introduced evidence related to its own licensing policy, though VirnetX never exclusively relied upon its policy alone to justify the royalty; instead it considered the policy in light of all the other *Georgia-Pacific* factors. *See* 11/01/12 p.m. TT at 103:8–19. As to the relevance of non-infringing alternatives, Apple agrees that these alternatives need to be feasible. *See Grain Processing Corp. v. American Maize-Prods. Co.*, 185 F.3d 1341, 1349 (Fed. Cir. 1999). Here, VirnetX presented evidence Apple's proposed alternatives were not. While FaceTime calls could go through relay servers, VirnetX established that routing all calls through these servers would not be a viable alternative, in light of the projected costs and downgrade in service. *See* 11/01/12 p.m. TT at 215:23–216:4; 11/01/12 a.m. TT at 140:4–143:19. In sum, VirnetX presented substantial evidence to support Mr. Weinstein's reasonable royalty rate of 1%.

3. <u>Nash Bargaining Solution</u>

Apple also attacks Mr. Weinstein's alternative damages theory, based on the Nash Bargaining Solution ("NBS"). Using its alternative damages model, VirnetX calculated that its damages for only the FaceTime feature would be $588 million. 11/01/12 p.m. TT at 136:11–146:19; 212:23–219:4. Apple acknowledges there are alternative methods of calculating reasonable royalties; however, it argues Mr. Weinstein did not use the generally accepted methods of applying NBS and thus his analysis is unreliable. Docket No. 623 at 35. Mr.

Weinstein based his calculations on the incremental profit associated with adding a front-facing camera, after equating the value of the front-face camera to the FaceTime feature. *Id.*; 11/01/12 p.m. TT at 218:4–11. Instead, Apple argues Mr. Weinstein should have determined the additional profits associated with the use of the technology. *Id.*; 11/01/12 p.m. TT at 138:12–14. Also, Apple contends Mr. Weinstein failed to explain why a 45%–55% profit split between VirnetX and Apple would have occurred. *Id.* at 35; *see* 11/01/12 p.m. TT at 141:13–17. Apple asserts this arbitrary profit split is akin to the disdained "25% rule," [8] and does not support a damages award since it has no basis in reality.

Mr. Weinstein adequately supported his alternative damages theory. First, he calculated the contribution of the "FaceTime" feature to Apple's total profits by estimating the price differential between the accused product and the last previous version of the product not capable of supporting the feature. 11/01/12 p.m. TT at 143:13–144:17; 219:1–6. Mr. Weinstein then reduced this revenue by the percentage of the revenue associated with the addition of the "FaceTime" feature. In this analysis, Mr. Weinstein relied on the price of the camera, because the addition of the camera enabled the feature. *See* 11/01/12 p.m. TT at 215:18-22. Additionally, Mr. Weinstein accounted for the 45%–55% profit split in his analysis, explaining that VirnetX would have been in a weaker bargaining position at the time of the negotiation because of its financial situation.[9] 11/01/12 p.m. TT at 216:5–217:4. Accordingly, Apple would have benefitted from this inequity in bargaining positions, and Mr. Weinstein modified the profit split

[8] Previously, the 25 percent rule had been used to approximate a reasonable royalty rate. However, the Federal Circuit has since held that "the 25 percent rule is a fundamentally flawed tool for determining a baseline royalty rate in a hypothetical negotiation. Evidence relying on the 25 percent rule of thumb is thus inadmissible under *Daubert* and the Federal Rules of Evidence, because it fails to tie a reasonable royalty base to the facts of the case at issue." *Uniloc USA, Inc.v. Microsoft Corp.*, 632 F.3d 1292,1312–13, 1315 (Fed. Cir. 2011).

[9] Traditionally, there is a 50-50 split of profits in the Nash Bargaining Solution. Apple first complains that Mr. Weinstein failed to properly explain his deviation from the traditional NBS approach. Apple's complaint is perplexing, as Weinstein's alleged error in this situation actually favors Apple. Apple then complains Mr. Weinstein failed to explain why he only reduced VirnetX's share to 45% as opposed to a lower figure, but Mr. Weinstein did explain why he chose to apply a 45-55 split. *See* 11/01/12 p.m. TT at 216:5–218:3.

to reflect this. Contrary to Apple's assertions, VirnetX did provide substantial evidence to supports its alternative damages theory.

4. Reduction of Jury Award

Apple also requests that the Court adjust the damages award, because there is no basis to award damages for any period prior to the time Apple received actual notice of the patents-in-suit. Docket No. 623 at 37. Apple argues VirnetX is not entitled to pre-litigation damages in this case because it failed to show Apple knew of the patents prior to the litigation. *Id.* If the Court finds VirnetX failed to show Apple knew of the patents prior to their assertion, Apple alleges that the jury had no information to limit their award, since Mr. Weinstein failed to produce any figures that subtracted pre-suit sales. *Id.* at 38–39. Therefore, the jury could not have rendered a "properly-limited damages award." *Id.* at 39.

However, VirnetX produced evidence that Apple did have pre-suit knowledge of the patents-in-suit. *See* 11/01/12 a.m. TT at 91:7–92:20 (Dr. Jones explaining that the VirnetX/Microsoft settlement was well publicized and when asked about this settlement, Apple could not deny someone knew about the license at Apple). Even assuming arguendo that VirnetX had failed to demonstrate Apple had actual knowledge of the patents prior to this litigation, VirnetX still produced figures that discounted pre-suit sales. Mr. Weinstein specifically explained that the damages at most would be reduced by $52 million. 11/01/12 p.m. TT at 131:16–134:17. Therefore, there is no basis to grant Apple's request to reduce the jury award.

Conclusion

In light of the foregoing, Apple's request for judgment as a matter of law in regards to damages is **DENIED.**

Motion for a New Trial or Remittitur

Apple argues, in the alternative, that it should be granted a new trial or remittitur, because: (1) the Court's jury instruction regarding damages was erroneous; (2) Mr. Weinstein's testimony should have been excluded or stricken; (3) the Court did not allow Apple to adequately rebut VirnetX's evidence of inducement; and (4) the verdict is against the greater weight of evidence. Docket No. 623 at 39–49. Apple also requests a new trial or remittitur if the Court finds the claims of either direct or induced infringement are not unsupported by substantial evidence. *Id.* As previously discussed, there was substantial evidence to support VirnetX's claims of direct and induced infringement; therefore, Apple's request with respect to findings of infringement is **DENIED.**

Jury Instruction

Apple argues the Court erroneously instructed the jury about the royalty base it could consider when awarding damages. Docket No. 623 at 39. The Court gave the following instruction:

> In determining a royalty base, you should not use the value of an entire apparatus or product unless **either:** (1) the patented feature creates the basis for customers' demand for the product, or the patented feature substantially creates the value of the other component parts of the product; or (2) the product in question constitutes the smallest saleable unit containing the patented feature.

Docket No. 597 at 27–28 (emphasis added). Apple contends this instruction allowed the jury to consider the entire value of the accused devices without complying with the entire market value rule. Docket No. 623 at 40–46. While Apple recognizes the concept of a "smallest saleable patent practicing unit," it argues this instruction runs afoul of decades of case law that states the entire value of a product may only be considered "[if] the patented feature creates the 'basis for customer demand' or 'substantially create[s] the value of the component parts.'" *Uniloc USA,* *Inc. v. Microsoft Corp.*, 632 F.3d 1292, 1318 (Fed. Cir. 2011) (citing *Lucent Tech., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1336 (Fed. Cir. 2009); Docket No. 632 at 41–45. Apple alleges the Court's error was extremely prejudicial, because it allowed the jury to consider an astronomically large royalty base, thus skewing the damages in VirnetX's favor. *Id.* at 45.

VirnetX argues that Apple again is conflating the entire market value rule and the smallest saleable patent-practicing unit. Docket No. 632 at 40–45. VirnetX does not argue with Apple about the requirements of the entire market value rule. Rather, VirnetX contends there are instances when a jury may consider the value of an entire product if the whole product is the smallest saleable unit with a "close relation to the claimed invention." *LaserDynamics, Inc. v. Quanta Comp. Inc.*, 694 F.3d 51, 67 (Fed. Cir. 2012); *see Broadcom Corp. v. Emulex Corp.*, No. 09-01058, 2011 U.S. Dist. LEXIS 154416 at *14 (C.D. Cal. Dec. 13, 2011)("The requirements of the entire market value rule must be met only if the royalty base is not the smallest saleable unit with close relation to the claimed invention.") Thus, if the smallest saleable unit is the product itself, then the entire market value rule should not be considered, since the rule is an exception that allows a jury to consider the entire revenues of a multicomponent product when the patented feature is only a small aspect of the product. Additionally, VirnetX refutes claims that the jury in fact considered the entire market value of the accused devices in this case, as the figure was never actually disclosed to the jury. Docket No. 632 at 45.

In this instance, the Court's instruction was not erroneous. While Apple is correct that the entire market value rule when applied requires a showing by plaintiff that the patented feature either drove demand for the entire product or substantially created the value of the component parts, the Court's instruction explained another alternative when the jury could consider the entire value of a product. There are instances when the smallest saleable patent-practicing unit is the entire product. Depending on the claim language of a patent, it is foreseeable that an entire product is required to practice the invention. *See Rite-Hite Corp. v. Kelley Co.*, 56 F.3d 1538, 1550 ("All the components together must be analogous to components of a single assembly or be parts of a complete machine, or they must constitute a functional unit. Our precedent has not extended liability to include items that have essentially no functional relationship to the patented invention and that may have been sold with an infringing device only as a matter of convenience or business advantage."). Accordingly, the Court's instruction to the jury was proper.

Mr. Weinstein's Testimony

Apple also reasserts its pre-trial *Daubert* motion and argues the Court erred in not excluding or striking Mr. Weinstein's testimony. Docket No. 623 at 46. To calculate VirnetX's damages, Mr. Weinstein applied several different damages models, including the Nash Bargaining Solution and a reasonably royalty rate based on a hypothetical negotiation. Apple contends Mr. Weinstein relied upon insufficient data and he incorrectly applied these damages models in calculating VirnetX's damages. *Id.* VirnetX disputes Apple's contentions and argues Mr. Weinstein's calculations were reliable and the Court did not err by admitting his testimony. Docket No. 632 at 46.

The Court has already addressed the sufficiency of Mr. Weinstein's data and the reliability of his alternative damages model. Additionally, prior to the trial, the Court made a preliminary determination that Mr. Weinstein's testimony was admissible because his opinions were well-reasoned and tied to the facts of the case, thus placing the burden on Apple to discredit his testimony at trial. *See Daubert v. Merrell Dow Pharm.*, 509 U.S. 579, 595 (1993) ("Vigorous cross–examination, presentation of contrary evidence, and careful instruction on the burden of proof are the traditional and appropriate means of attacking shaky but admissible evidence.") Accordingly, it was proper for Mr. Weinstein to testify.

Evidentiary Rulings

Apple also argues it is entitled to a new trial because the Court erred in allowing VirnetX to present testimony from Apple engineers in order to establish Apple's liability for inducement, without affording Apple the opportunity to discuss the pending re-examinations at the USPTO. Docket No. 623 at 47–48. Apple first argues the Court erred by allowing VirnetX to present the testimony of Apple employees, since this evidence gave the incorrect impression that Apple as a company disregards others' intellectual property. However, Apple has not demonstrated it was an error to admit this evidence. Contrary to Apple's assertion, this evidence was relevant to VirnetX's contention that Apple took steps to avoid confirming infringement. From this evidence, the jury could have inferred Apple as a company actively avoids investigating the scope of prior art when it is developing a "new" application. While Apple argues this evidence should have been excluded because it was prejudicial, Apple has failed to explain why this concern outweighs the probative value of the evidence. *See* FED. R. EVID. 403.

Apple also contends the Court erred when it prevented Apple from refuting VirnetX's contentions, with evidence Apple initiated re-examinations at the USPTO to contest the validity of the patents-in-suit. *Id.* Apple argues this error was further compounded by the Court's instruction that the jury could consider "whether or not Apple obtained the advice of a competent lawyer" or "whether or not it knew of the patents or was willfully blind to the patents when designing and manufacturing its products." Docket No. 597 at 17. Without evidence that Apple in fact took steps after learning of VirnetX's allegations of patent infringement, Apple argues the jury had an incomplete and flawed understanding of the facts, which undoubtedly influenced its consideration of VirnetX's claim of induced infringement. Docket No. 623 at 48.

It is within the purview of the Court to exclude highly prejudicial evidence that risks misleading the jury. *Id.* Here, the re-examinations were still ongoing, meaning there was no final determination by the USPTO with respect to the validity of the patents-in-suit. If evidence of the ongoing re-examinations had been allowed, it could have improperly influenced the jury's decision regarding validity. *See Callaway Golf Co. v. Acushnet Co.*, 576 F.3d 1331, 1342 (Fed. Cir. 2009)(finding the district court did not err in excluding evidence of a pending re-examination, noting the risk of jury confusion was high if such evidence was introduced). Apple was free to bring in other evidence that did not risk misleading the jury as to other trial issues to combat VirnetX's allegations of willful blindness. Apple has not established it was an error to exclude evidence of the re-examinations.

Weight of the Evidence

Lastly, Apple summarily asks the Court to exercise its discretion and grant a new trial, because the great weight of the evidence supports Apple's contentions that it does not infringe the asserted claims, these claims are invalid, and the damages award is unsupported. For reasons previously stated, the great weight of the evidence does not support Apple's arguments.

Conclusion

In light of the foregoing, Apple's alternative motion for a new trial or remittitur is **DENIED.**

VIRNETX'S MOTION FOR POST-VERDICT DAMAGES TO THE TIME OF JUDGMENT, PRE-JUDGMENT INTEREST, AND POST-JUDGMENT INTEREST and VIRNETX'S AMENDED MOTION FOR POST-VERDICT DAMAGES

In light of the jury's verdict, VirnetX now requests pre-judgment and post-judgment interest, and post-verdict damages. Docket No. 620 at 1. Apple does not oppose awarding

VirnetX pre-judgment interest, however it disputes which interest rate should be awarded. Docket No. 630 at 2–4. Additionally, Apple does not contest that VirnetX is entitled to post-judgment interest pursuant to 28 U.S.C. § 1961 (a). *Id.* at 4. However, the parties dispute whether VirnetX should be awarded post-verdict damages.

Pre-judgment Interest

Apple does not oppose awarding VirnetX pre-judgment interest, however it disputes which interest rate should be used. Docket No. 630 at 2–4. VirnetX asks the Court to award the Texas statutory interest rate of 5%, compounded quarterly, while Apple has requested the Court award the periodic prime interest rate of 3.25%, compounded quarterly. Docket No. 620 at 5; Docket No. 638 at 2. In keeping with the standard practice of this Court, VirnetX shall be awarded pre-judgment interest on the $368,160,000.00 damage award at the prime rate, compounded quarterly, such that the pre-judgment interest is $8,755,381 through October 31, 2012, with daily interest of $33,561. *See Soverain Software LLC v. J.C. Penney Corp.*,No. 6:09-cv-272, 2012 U.S. Dist. LEXIS 151102, *33 (E.D. Tex. Aug. 9, 2012); *Clear With Computers, LLC v. Hyundai Motor Am., Inc.*, No. 6:09-cv-479, slip op. at 14 (E.D. Tex. Jan. 9, 2012); *Fractus, S.A. v. Samsung Elecs. Co.*, 876 F. Supp. 2d 802, 2012 U.S. Dist. LEXIS 80284, *143 (E.D. Tex. 2012); *Tele-Cons, et al. v. General Electric Co, et al.*, No. 6:10-cv-451, slip op. at 2 (E.D. Tex. Sept. 29, 2011); *ACQIS LLC v. IBM Corp. et al.*, No. 6:09-cv-148, slip op. at 1 (E.D. Tex. June 8, 2011).

Post-Verdict Damages

Apple asks the Court to deny VirnetX's request for post-verdict damages, because it contends VirnetX has grossly overestimated its damages Docket No. 630 at 1. In its original request, VirnetX had extrapolated from Apple's disclosed sales figures to determine the number of sales of accused products post-verdict. Docket No. 620 at 8. Apple contends VirnetX's post-verdict damages model overestimates the number of sales post-verdict, because VirnetX improperly assumed the sale of accused products would remain consistent throughout the year. Docket No. 630 at 2. Apple argues this is incorrect, because its release of new products in late 2012 would have decreased the demand for the accused products. Historically, when Apple released a new model, sales of the prior iteration would decline. Therefore, Apple contends VirnetX's request is too high.

At the post-trial hearing, VirnetX emphasized it only had sales information up to June 2012 and therefore could not determine the impact of new products on accused product sales. In light of the ongoing dispute as to how the release of new products would actually affect accused product sales, the Court requested additional briefing on the matter. Docket No. 647. The Court ordered Apple to produce its most current sales data up to December 31, 2012 and ordered VirnetX to file an amended request in light of Apple's new sales figures.

VirnetX amended its request in light of the new data to $330,201 per day from November 6, 2012 through the date of the final judgment. Docket No. 657 at 2. Apple still disputes this figure, claiming VirnetX again did not sufficiently discount the post-verdict damages to reflect the decrease in accused product sales. Docket No. 664 at 2. Apple instead proposes awarding post-verdict damages of $279,229 a day.[10] *Id.*

Upon further review of VirnetX's and Apple's competing proposals, the Court rejects Apple's proposal. Essentially Apple is asking VirnetX to look into a crystal ball to divine if and how accused product sales will further decrease. VirnetX was only given sales information up to

[10] Apple estimated the decrease in accused product sales by averaging the decrease in sales of other older models after the release of new products. However, the percentage change varied widely ranging, from less than -5% to over -60%. It is uncertain at this moment whether the accused products in this case would experience such a significant shift or not.

December 31, 2012, and VirnetX's methodology is reasonable in light of the almost impossible task Apple is now requesting.

Conclusion

VirnetX's request for pre-judgment interest and post-judgment interest is **GRANTED IN PART** and **DENIED IN PART.** Additionally, VirnetX's amended motion for post-verdict damages is **GRANTED.**

VIRNETX'S MOTION FOR PERMANENT INJUNCTION

VirnetX also requests an order permanently enjoining Apple from infringing the patents-in-suit. Docket No. 621 at 1.[11]

Applicable Law

In determining whether to issue a permanent injunction in patent cases, courts apply the four factor test provided for in *eBay, Inc. v. MercExchange, LLC*, 547 U.S. 388, 394 (2006). A party is entitled to a permanent injunction only if: "(1) [the party] has suffered an irreparable

[11] VirnetX proposed the following language: Apple, Inc., its officers, agents, servants, employees, and attorneys, and those persons in active concert or participation with them who receive actual notice of the order by personal service or otherwise are permanently restrained and enjoined from performing the following actions during the term of U.S. Patent Nos. 6,502,135 and 7,490,151: 1. making, offering to sell, selling, importing, and/or using VPN on Demand Infringing and Future Products in or into the United States; 2. instructing or encouraging anyone (alone or jointly with Apple) to make, offer to sell, sell, import, and/or use in or into the United States VPN on Demand Infringing and Future Products; and 3. selling a component with a capability identical to or not more than colorably different from the capability of determining whether to initiate a VPN based on comparing a domain name to a list of domain names. Notwithstanding the foregoing, Apple is not enjoined from performing the above actions with respect to VPN on Demand Infringing and Future Products that were included in the royalty base at trial or covered by the Court's *per diem* through judgment. Apple, Inc., its officers, agents, servants, employees, and attorneys, and those persons in active concert or participation with them who receive actual notice of the order by personal service or otherwise are permanently restrained and enjoined from performing the following actions during the term of U.S. Patent Nos. 7,418,504 and 7,921,211: 1. making, offering to sell, selling, importing, and/or using one or more FaceTime Infringing and Future Servers in or into the United States; 2. instructing or encouraging anyone, alone or jointly with Apple, to make, offer to sell, sell, import, and/or use (including by distributing computer code or a FaceTime Infringing and Future Product that includes the capability of using) in or into the United States one or more FaceTime Infringing and Future Servers; 3. instructing or encouraging anyone, alone or jointly with Apple, to use (*e.g.*, use by a FaceTime Infringing and Future Product) one or more FaceTime Infringing and Future Servers that are in the United States; and 4. selling a component with a capability identical to or not more than colorably different from the capability of supporting establishing a non-relayed, encrypted FaceTime call. Notwithstanding the foregoing, Apple is not enjoined from supporting establishment of non-relayed, encrypted FaceTime calls between two or more FaceTime Infringing and Future Products that were all included in the royalty base at trial or covered by the Court's *per diem* through judgment. Docket No. 621, Ex. 5 at 2–4.

injury; (2) that remedies at law, such as monetary damages, are inadequate to compensate for that injury; (3) that, considering the balance of hardships between the [parties], a remedy in equity is warranted; and (4) that the public interest would not be disserved by a permanent injunction." *Id.* at 391. The Supreme Court held "the decision whether to grant or deny injunctive relief rests within the equitable discretion of the district courts, and that such discretion must be exercised consistent with traditional principles of equity, in patent disputes no less than in other cases governed by such standards." *Id.*

Irreparable harm and monetary damages

VirnetX argues Apple's infringement irreparably harmed the company, because Apple's infringement prevented VirnetX from releasing its own technology, the Gabriel technology. Docket No. 621 at 4. VirnetX has been developing this technology, which practices the claimed invention and in VirnetX's opinion, could be implemented by Apple to replace Apple's infringing software and servers. *Id.* at 4, 6; *see* 10/31/12 a.m. TT at 119:9:123–1. VirnetX contends that Apple's infringement prevented the company's entrance into the market, because Apple's infringing products have saturated the market, thus making it impossible for VirnetX to compete. *Id.* at 4–5.

Additionally, Apple's decision to infringe VirnetX's patents instead of licensing the Gabriel technology prevented the implementation of VirnetX's technology on millions of devices. *Id.* at 6. Specifically, VirnetX asserts an agreement between Apple and VirnetX would have given VirnetX a "beachhead in the industry." *Id.* This "beachhead" would have allowed VirnetX to "scale [its] infrastructure," "hire more developers," and provided VirnetX "a venue for participating in this technology and seeing it go out into the – into the marketplace." *Id.* at 6 (quoting 10/31/12 a.m. TT at 118:17–119:4).

Apple counters that VirnetX has not been irreparably harmed, because VirnetX has misrepresented the nature of the Gabriel technology, and VirnetX has not demonstrated its failure to succeed in the market is a result of Apple's alleged infringement. Docket No. 631 at 2–7. Apple disputes the relevance of VirnetX's Gabriel technology in the analysis, since the product is not even commercially available. *Id.* at 4. In the absence of any commercially available product, Apple contends VirnetX cannot allege it has been irreparably harmed, because VirnetX does not compete with Apple products. *Id.* at 3–4. Without a product, VirnetX cannot allege it has lost market share, brand recognition, or customer goodwill. *Id*. at 3. Even if VirnetX did have a product, Apple contends VirnetX would not directly compete with Apple, since Apple sells hardware whereas products implementing the patents-in-suit would cover software. *Id.* at 3–4.

Additionally, Apple argues VirnetX has failed to prove Apple's infringement affected VirnetX's presence in the marketplace. First, Apple contends there is no evidence that demonstrates Apple's infringement prevented VirnetX's entrance into the market. Docket No. 631 at 5. VirnetX has acknowledged that its Gabriel technology is capable of running on other operating systems, such as the Android operating system. *Id.* at 6; *See* 10/31/12 a.m. TT at 120:2–6; 120:16–121:5; 10/31/12 p.m. TT at 10:12–18. However, there is nothing in the record to suggest VirnetX was unable to sell its technology or license its patents to other smartphone, tablet or computer manufacturers, because of Apple's infringement. *Id.* at 5–6; *see* 10/31/12 a.m. TT at 130:12–132:12. In fact, VirnetX entered licensing agreements after June 2009, the time when Apple began selling the accused products. *Id.* at 5.

Apple also argues VirnetX has failed to demonstrate that Apple's infringement deprived VirnetX of a significant business opportunity. Docket No. 631 at 7. Apple argues that VirnetX has improperly assumed that Apple would have chosen to license the Gabriel technology in 2009 instead of implementing a design–around or opting to license the patents–in–suit. *Id*. However, there is nothing to suggest that Apple would have chosen to license the Gabriel technology, especially considering at the time Apple began selling the accused products VirnetX's technology was incapable of running on many Apple devices. *Id.*, 10/31/12 p.m. TT at 9:10–10:13 (Dr. Short, one of the inventors of the patents–in–suit, acknowledging that the Gabriel technology does not run on the iOS operating system).

VirnetX's Gabriel technology is currently unavailable commercially. *See* 10/31/12 p.m. TT at 11:15–12:11 (Dr. Short stating there are no commercial implementations of the Gabriel technology currently, no companies are running their own beta testing of the product, and there has only been internal testing of the product). Even assuming VirnetX's technology was available for purchase, Apple does not directly compete with VirnetX. Apple sells phones, not security software. Additionally, Apple's sale of cell phones has not restricted VirnetX's ability to market and sell its Gabriel technology to other tablet, cellphone or computer manufacturers. VirnetX's damages are limited to the loss of Apple as a customer.

VirnetX has also argued the loss of Apple as a licensee further impacted its ability to establish itself in the marketplace. While evidence of past harm can be considered by the Court, the Court cannot engage in counterfactual thinking and determine what might have been. *See i4i Ltd. P'ship v. Microsoft Corp.*, 598 F.3d 831, 861 (Fed. Cir. 2010). Assuming Apple could have implemented VirnetX's technology for all the accused products, and assuming Apple would have decided to license the technology, one cannot say this would have guaranteed VirnetX's success. In absence of any evidence, Apple cannot be held solely responsible for VirnetX's failure to obtain a foothold in the market.

Here, Apple's infringement has caused VirnetX to lose Apple as a licensee, which can be remedied with monetary damages. VirnetX is not excluded from selling or licensing its products to other companies in the market and has not suffered any loss of good will, lost profits or market share. Accordingly, VirnetX has not demonstrated that it will suffer irreparable harm absent a permanent injunction and that monetary damages are an insufficient remedy.

Permanent Injunction: Balance of Hardships and Public Interest

VirnetX has not demonstrated that Apple's infringement is responsible for VirnetX's failure to enter and be successful in the marketplace. Apple's sale of infringing products has not restricted VirnetX's ability to approach other companies or sell its products to consumers. Additionally, while a licensing agreement between Apple and VirnetX may have boosted VirnetX's position in the market, there is no evidence demonstrating that VirnetX lost goodwill because of Apple's infringement.

With respect to Apple, while throughout the trial it attempted to minimize the importance of the infringing features in its products, Apple still bears a considerable burden to comply with the proposed injunction. The most recent estimates project the cost to comply at $50.8 million. Docket No. 653, Ex. 2 at 2; *Id.*, Ex. 3 at 3. Additionally, though VirnetX only seeks to enjoin the use of the infringing feature and not the entire devices, an injunction would not only harm Apple, but also its customers and other third parties. Although the Court is concerned about the great disparity between Apple's proposed costs to comply with an injunction presented at trial and those presented post-trial, the balance of hardships weigh against enjoining Apple.

Conclusion: Permanent Injunction

Accordingly, VirnetX's Motion for a Permanent Injunction is **DENIED.**

<u>*Ongoing Royalty*</u>

If the Court does not grant a permanent injunction against Apple, VirnetX "requests that the Court order the parties to negotiate a license regarding Apple's future use of VirnetX's patented technology." Docket No. 621 at 18. If the parties fail to reach an agreement, VirnetX then asks that Court to impose an ongoing royalty. *Id.* The Federal Circuit has encouraged courts to allow the parties to negotiate a license amongst themselves regarding the future use of a patented technology prior to the court imposing a royalty. *See Paice LLC v. Toyota Motor Corp.*, 504 F.3d 1293, 1315 (Fed. Cir. 2007); *Telecordia Techs., Inc. v. Cisco Sys., Inc.*, 612 F.3d 1365, 1378–79 (Fed. Cir. 2010).

Accordingly, to provide finality to the trial, the Court **SEVERS** VirnetX's claim for an ongoing royalty into a separate cause of action. *See* FED.R.CIV.P. 21. ("[t]he court may sever any claim against a party."). The parties are **ORDERED** to meet and confer to schedule mediation within 45 days of this order to attempt to negotiate a license. Should the parties fail to agree regarding Apple's future use of VirnetX's patents at the mediation, VirnetX is **ORDERED** to file the appropriate motion.

VIRNETX'S MOTION FOR ENTRY OF JUDGMENT, REQUEST FOR ATTORNEYS' FEES, AND JUDGMENT AGAINST APPLE ON APPLE'S LATE-ABANDONED COUNTERCLAIMS AND DEFENSES

VirnetX requests attorneys' fees in light of Apple's litigation misconduct and the *Read* factors. VirnetX also asks for entry of judgment against Apple and specifically requests judgment against Apple on Apple's defenses and counterclaims that were pending at the time of trial, but were not actually presented to the jury.

<u>*Attorneys' Fees*</u>

Attorneys' fees and costs may be awarded in "exceptional cases" to the "prevailing party." 35 U.S.C. § 285. Awarding attorneys' fees under § 285 is a two-step process. *Cybor Corp. v.*

FAS Tech., 138 F.3d 1448, 1460 (Fed. Cir. 1998). "First, the district court must determine whether a case is exceptional, a factual determination reviewed for clear error. After determining that a case is exceptional, the district court must determine whether attorney fees are appropriate." *Id.* (internal citation omitted); *Delta-X Corp. v. Baker Hughes Prod. Tools, Inc.*, 984 F.2d 410, 413 (Fed. Cir. 1993).

"Exceptional cases usually feature some material, inappropriate conduct related to the matter in litigation, such as willful infringement, fraud or inequitable conduct in procuring the patent, misconduct during litigation, vexatious or unjustified litigation, conduct that violated Federal Rule of Civil Procedure 11, or like infractions." *Serio-US Indus., Inc. v. Plastic Recovery Techs. Corp.,* 459 F.3d 1311, 1321–22 (Fed. Cir. 2006). "[E]xceptional cases" may, but are not required to, include the jury's finding of willfulness. *Insituform Techs., Inc. v. Cat Contracting, Inc.*, 518 F. Supp. 2d 876, 895 (S.D. Tex. 2007) (citing *Avia Group Int'l, Inc. v. L.A. Gear Ca., Inc.*, 853 F.2d 1557, 1567 (Fed. Cir. 1998)). For example, a case may be exceptional based solely on litigation misconduct and unprofessional behavior. *Rambus, Inc. v. Infineon Techs. AG*, 318 F.3d 1081, 1106 (Fed. Cir. 2003); *Epcon Gas Sys., Inc. v. Bauer Compressors, Inc.*, 279 F.3d 1022, 1034 (Fed. Cir. 2002). Ultimately, "[t]he decision to increase damages is committed to the discretion of the trial judge." *Modine Mfg. Co. v. Allen Group, Inc.*, 917 F.2d 538, 543 (Fed. Cir. 1990).

VirnetX argues Apple's litigation misconduct and the application of some *Read* factors make this case exceptional. Docket No. 625 at 2, 11. Specifically, VirnetX claims Apple disregarded the Court's orders on several occasions, urged meritless defenses, concealed its primary infringement defense until the eve of trial, contradicted prior representations it made to the Court, failed to comply with the local rules during discovery, and generally acted like a knave. *Id.* at 3–11. Apple disputes VirnetX's claims, alleging VirnetX greatly exaggerates and misrepresents Apple's behavior. Docket No. 629 at 1.

Although, Apple failed to always act as professional as this Court expects[12], its "misconduct" did not rise to the level that warrants the award of attorneys' fees. *Cf. ReedHycalog UK, Ltd. v. Diamond Innovations, Inc.*, No. 6:08-cv-325, 2010 U.S. Dist. LEXIS 83011, *23 (E.D. Tex. Aug. 12, 2010) (finding a case exceptional when the defendant repeatedly acted in bad faith and attempted to conceal "relevant and discoverable but damaging documents"); *z4 Techs., Inc. v. Microsoft Corp.,* No. 6:06-cv-142, 2006 U.S. Dist. LEXIS 58374, *68–76 (E.D. Tex. Aug. 18, 2006)(awarding attorneys' fees when the defendant withheld critical evidence, took intentional steps to mislead the Court on several occasions, and attempted to conceal relevant trial evidence with its voluminous exhibit tactic). There is little, if any, evidence that Apple intentionally disregarded this Court's orders or explicitly contradicted statements it made to the Court. Apple was guilty of misconduct when it improperly terminated a deposition, however, Apple has already been sanctioned for this behavior. *See* Docket No. 450. As to Apple's other incidents of "egregious" behavior, it does not appear that Apple acted in bad faith. At best, VirnetX cites to some occasions of rude behavior, but this alone does not support declaring this to be an exceptional case.

Additionally, VirnetX urges the Court to award attorneys' fees because of Apple's size and financial condition, the duration of Apple's misconduct, Apple's failure to take remedial action, and the outcome of the case. *Id.* at 11–13. The application of the *Read* factors in the absence of a finding of willful infringement does not support declaring this an exceptional case. Accordingly, the Court **DENIES** VirnetX's request for attorneys' fees.

[12] This Court understands the litigious nature of these proceedings; however, this does not excuse a party from acting with the courtesy and respect one should afford a fellow professional.

Judgment on Defenses and Counterclaims not Presented at Trial

VirnetX also requests the Court to enter judgment on all of Apple's invalidity defenses and counterclaims, including Apple's prior art references, which Apple asserted up to the time of trial, but never presented to the jury. Docket No. 625 at 13. VirnetX contends judgment should be entered because there was still an actual controversy as to the validity of every claim of the patents-in-suit at the time of trial.

Initially, VirnetX asserted a substantial number of claims, which it narrowed to 77 asserted claims at the time of the pre-trial hearing. *See* Docket No. 482. Apple itself initially asserted several defenses and counterclaims, including anticipation, obviousness, and failure to comply with the written description requirement, however as VirnetX narrowed its claims so too did Apple narrow its defenses and counterclaims. VirnetX concedes it only presented 16 claims to the jury, however, it argues it never actually dismissed any of its asserted claims. *Id.* at 14.[13] Rather, it merely narrowed its case to present to the jury.

In patent cases, "the existence of a case or controversy must be evaluated on a claim-by-claims." *Jervis B. Webb Co. v. So. Sys., Inc.*, 742 F.2d 1388, 1399 (Fed. Cir. 1984). Additionally, the existence of a case or controversy "must exist at all stages of review, not merely at the time the complaint [was] filed." *Streck, Inc. v. Research & Diagnostic Sys.*,665 F.3d 1269, 1282 (Fed. Cir. 2012)(original citations omitted). Thus, there must be a showing that a "continuing case or controversy with respect to withdrawn or otherwise unasserted claims" still exists. *Id.* at 1283.

Here, Apple only presented an anticipation defense to the jury, and only with respect to the claims asserted by VirnetX at trial. Apple presented no other evidence at trial regarding its

[13] Only claims 1, 3, 7, 8 of the '135 Patent; claims 1 and 13 of the '151 Patent; claims 1, 2, 5, 16, 21, and 27 of the '504 Patent; and claims 36, 37, 47 and 51 of the '211 Patent were asserted by VirnetX at trial.

other invalidity theories and prior art references. The Court cannot and will not enter judgment upon claims and defenses that were not presented for consideration to the jury. There is no basis to enter such a judgment, no more than there is a basis to enter judgment of non-infringement for Apple as to VirnetX's unasserted claims. *See Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1553 (Fed. Cir. 1983); *Datascope Corp. v. Smec, Inc.*, 776 F.2d 320, 327 (Fed. Cir. 1985)(noting that references in pleadings do not support a judgment that particular claims are invalid when "the validity of those claims were not litigated by the parties at trial"); *Nordisk Pharm., Inc. v. Bio-Technology Gen. Corp.*, 424 F.3d 1347, 1356 (Fed. Cir. 2005) (vacating the district court's order invalidating a claim not litigated at trial). The Court encourages and requires the parties to narrow their case for trial. Accordingly, the Court will not penalize such attempts to narrow issues by entering judgment on issues not presented at trial.

Conclusion

In light of the foregoing, VirnetX's Motion for Entry of Judgment on the Jury Verdict, Request for Attorneys' Fees, and Judgment against Apple on Apple's Late-Abandoned Counterclaims and Defenses, including all of Apple's Alleged Prior Art References is **GRANTED IN PART** and **DENIED IN PART.** The Court will enter judgment against Apple only on the claims presented at trial.

CONCLUSION

Accordingly, Apple's Motion for Judgment as a Matter of Law under Rule 50(b) or, in the alternative, for a New Trial or a Remittitur is **DENIED**. VirnetX's Motion for Post-Verdict Damages to the Time of Judgment, Pre-Judgment interest, and Post-Judgment Interest is **GRANTED IN PART** and **DENIED IN PART**. VirnetX's Amended Motion for Post-Verdict Damages is **GRANTED.** VirnetX's Motion for Permanent Injunction is **DENIED**, and

SEVERS VirnetX's request for an Ongoing Royalty into a separate action. Lastly, VirnetX's Motion for Entry of Judgment on the Jury Verdict, Request for Attorneys' Fees, and Judgment against Apple on Apple's Late-Abandoned Counterclaims and Defenses, including all of Apple's Alleged Prior Art References is **GRANTED IN PART** and **DENIED IN PART**.

So ORDERED and SIGNED this 26th day of February, 2013.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

TAB 4

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

	§	
VIRNETX INC.,	§	
	§	
Plaintiff,	§	
	§	
vs.	§	**CASE NO. 6:10-CV-417**
	§	
CISCO SYSTEMS, INC. et al.,	§	
	§	
Defendants.	§	
	§	

FINAL JUDGMENT PURSUANT TO FED. R. CIV. P. 54(b)

On August 11, 2010, VirnetX, Inc. ("VirnetX") filed this action against Apple, Inc. ("Apple") and several other parties alleging patent infringement. On the eve of trial, only two parties remained: Apple and Cisco Systems, Inc. By agreement, the Court granted Defendants' Motion for Separate Trials, setting only Apple for trial starting October 31, 2012 (Docket No. 542). Since all issues, between VirnetX and Apple, except future ongoing royalties, if any, have been finally resolved either by the jury or the Court's Memorandum Opinion and Order (Docket No. 732), there is no reason to delay entering judgment as to Apple.

Therefore, pursuant to Rule 54(b) of the Federal Rules of Civil Procedure, consistent with the Court's Memorandum Opinion and Order, and the Court having expressly determined that there is no just cause for delay, the Court **ORDERS AND ENTERS FINAL JUDGMENT AS TO APPLE,** as follows:

- Defendant Apple is found to infringe claims 1, 3, 7, 8 of U.S. Patent No. 6,502,135; claims 1 and 13 of U.S. Patent No 7,490,151; claims 1, 2, 5, 16, 21, and 27 of U.S. Patent No 7,418,504; and claims 36, 37, 47 and 51 of U.S. Patent No. 7,921,211 (collectively, "the Asserted Claims").

- The Asserted Claims are valid.
- The Court awards damages to VirnetX for Apple's infringement of the Asserted Claims in the amount of $368,160,000.
- VirnetX is further awarded pre-judgment interest, post-judgment interest, and post-verdict damages as detailed in the Court's Memorandum Opinion and Order.

All relief not specifically granted herein is **DENIED**, subject to **SEVERANCE** of VirnetX's request for an ongoing royalty as ordered in the Court's Memorandum Opinion and Order.

All pending motions between VirnetX and Apple not previously resolved, specifically: VirnetX's Opposed Motion *In Limine* (Docket No. 538); Apple's Opposed Motion *In Limine* (Docket No. 539); Apple's Motion to Dismiss for Lack of Jurisdiction (Docket No. 589); Apple's Judgment as a Matter of Law (Docket No. 593); VirnetX's Judgment as a Matter of Law (Docket No. 594); and VirnetX's Motion for Entry of Judgment (Docket No. 622) are **DENIED.**

So ORDERED and SIGNED this 27th day of February, 2013.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

TAB 5

**IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION**

	§	
VIRNETX INC.,	§	
	§	
Plaintiff,	§	
	§	
vs.	§	**CASE NO. 6:10-CV-417**
	§	
CISCO SYSTEMS, INC. et al.,	§	
	§	
Defendants.	§	
	§	

ORDER

Before the Court is Apple Inc.'s Motion to Alter or Amend the Judgment (Docket No. 793). For the reasons stated below, the Motion is **DENIED.**

BACKGROUND

On August 11, 2010, VirnetX, Inc. ("VirnetX") filed this action against Apple Inc. ("Apple") alleging that Apple infringed U.S. Patent Nos. 6,502,135 ("the '135 Patent"), 7,418,504 ("the '504 Patent), 7,490,151 ("the '151 Patent), and 7,921,211 ("the '211 Patent) (collectively, "the patents-in-suit"). The '135 and '151 Patents generally describe a method of transparently creating a virtual private network ("VPN") between a client computer and a target computer, while the '504 and '211 Patents disclose a secure domain name service.

A jury trial regarding the instant suit commenced on October 31, 2012. Following a five-day trial, the jury returned a verdict that the '135, '151, '211, and '504 Patents were not invalid and Apple infringed the asserted claims.[1] Docket No. 598. After addressing the parties' post-trial motions, the Court entered a final judgment against Apple on February 28, 2013. Docket

[1] At trial, VirnetX contended Apple infringed claims 1, 3, 7, 8 of the '135 Patent; claims 1 and 13 of the '151 Patent; claims 1, 2, 5, 16, 21, and 27 of the '504 Patent; and claims 36, 37, 47 and 51 of the '211 Patent.

Nos. 732, 742. The Court also severed VirnetX's claim for an ongoing royalty into a separate action and ordered the parties to attempt to negotiate a license for a post-verdict royalty without the Court's intervention. *See* Docket No. 732 at 42.

APPLICABLE LAW

A motion to alter or amend a judgment under Federal Rule of Civil Procedure 59(e) "serve[s] the narrow purpose of allowing a party to 'correct manifest errors of law or fact or to present newly discovered evidence.'" *Templet v. Hydrochem Inc.*, 367 F.3d 473, 479 (5th Cir. 2004) (quoting *Waltman v. Int'l Paper Co.*, 875 F.2d 468, 473 (5th Cir. 1989)). "[S]uch a motion is not the proper vehicle for rehashing evidence, legal theories, or arguments that could have been offered or raised before the entry of judgment." *Id.* (citing *Simon v. United States*, 891 F.2d 1154, 1159 (5th Cir. 1990)). This legal standard "favor[s] the denial of motions to alter or amend a judgment." *S. Constructors Grp., Inc. v. Dynalectric Co.*, 2 F.3d 606, 611 (5th Cir. 1993).

ANALYSIS

Apple asks the Court to amend or alter the final judgment. Docket No. 793. Specifically, Apple requests that the Court vacate the portion of the judgment severing VirnetX's request for an ongoing royalty into a new case, reclassify the judgment as non-final, and amend the judgment after the disposition of VirnetX's ongoing royalty request. *Id.* at 1. Apple argues it was improper for the Court to sever the claim for an ongoing royalty, because an ongoing royalty is not a separate cause of action but rather merely a remedy. *Id.* Since the severance was improper, Apple contends the ongoing royalty issue is still pending in the original case, therefore the judgment is not final.

VirnetX counters that the judgment should not be amended, because there are no outstanding issues in the instant case – liability and damages have been determined. Docket No.

802 at 7–8. VirnetX argues Apple misconstrues the purpose of an ongoing royalty as merely a remedy and not a separate cause of action. *Id.* at 2–7. An ongoing royalty addresses ongoing infringement. Since ongoing infringement is different from past infringement, VirnetX contends it was proper for the Court to sever the ongoing royalty from the instant case, thus the judgment should not be amended. *Id.*

Essentially, Apple argues that because there is only one claim of patent infringement at issue, there is no second claim that can be severed from the case. However, Apple fails to acknowledge that patent infringement is not a singular injury felt by the patentee, but may be an ongoing harm, if the infringer continues to make, use, offer to sell, or sells the patented invention during the term of the patent. 35 U.S.C. § 271 (2011). When VirnetX originally filed this action, it not only alleged past infringement but also ongoing infringement.[2] While Apple is correct that an ongoing royalty is a remedy, it is a remedy for ongoing infringement post-judgment, which is distinct from past infringement. Due to the realities of litigation, there can be separate claims for infringement: pre-judgment and post-judgment infringement. *See Paice LLC v. Toyota Motor Corp.*, 504 F. 3d 1293, 1316–17 (Fed. Cir. 2007) (Rader, R., concurring) ("Evidence and argument on royalty rates were, of course, presented during the course of the trial, for the purposes of assessing damages for Toyota's *past* infringement. But pre-suit and post-judgment acts of infringement are distinct, and may warrant different royalty rates given the change in the parties' legal relationship and other factors.") Thus, there were multiple claims present in this case which could be severed.

[2] *See* Docket No. 1 at 7 ("Apple has infringed and/or continues to infringe the '135 and '151 patents); Docket No. 75 at 9 ("Apple has infringed and/or continues to infringe the '135, '151 and '504 patents); Docket No. 107 at 11 ("Apple has infringed and/or continues to infringe the '135, '151, '504 and '211 patents); Docket No. 238 at 12 ("Apple has infringed and/or continues to infringe the '135, '151, '504 and '211 patents).

Rule 21 states "[t]he court may sever any claim against a party." FED. R. CIV. P. 21. "Severance under Rule 21 creates two separate actions or suits where previously there was but one. Where a single claim is severed out of a suit, it proceeds as a discrete, independent action, and a court may render a final, appealable judgment in either one of the resulting two actions notwithstanding the continued existence of unresolved claims in the other." *United States v. O'Neil*, 709 F.2d 361, 368 (5th Cir. 1983). "If a severed claim is one of multiple claims asserted by a single plaintiff against a single defendant (or vice versa), neither party is severed from the original action. Rather, both parties are involved in two separate actions." 4 JAMES WM. MOORE, ET AL., MOORE'S FEDERAL PRACTICE § 21.06 (3d ed. 2013).

A trial court has great discretion to sever the issues "to be tried before it." *Brunet v. United Gas Pipeline Co.*, 15 F.3d 500, 505 (5th Cir. 1994); *see* FED. R. CIV. P. 21. When a party requests an ongoing royalty, the Court believes severing the action for ongoing infringement is the most efficient way to handle the ongoing dispute. *See z4 Techs., Inc. v. Microsoft Corp.*, 434 F. Supp. 2d 437, 444 (E.D. Tex. 2006) (severing z4's continuing causes of action regarding Microsoft's post-verdict infringement into a separate action); *see also Mondis Tech. Ltd. v. Chimei Innolux Corp.*, 822 F. Supp. 2d 639, 642 (E.D. Tex. 2011); *LaserDynamics, Inc. v. Quanta Computer, Inc.*, No. 2:06-cv-348, Docket No. 825 (E.D. Tex. June 3, 2011); *Avid Identification Sys., Inc. v. Phillips Elecs. N. Am. Corp.*, No. 2:04-cv-183, 2008 WL 819962, at *4 (E.D. Tex. March 25, 2008); *Saffran v. Boston Scientific Corp.*, No. 2:05-cv-547, 2008 U.S. LEXIS 106711, at *2 (E.D. Tex. Feb. 14, 2008); *Alexsam Inc. v. IDT Corp.*, No. 2:07-cv-420, Docket No. 367 (E.D. Tex. Aug. 18, 2011). Prior to the Court deciding whether to impose a royalty for the future use of a patented technology, parties are afforded an opportunity to negotiate a license themselves. *See Paice LLC v. Toyota Motor Corp.*, 504 F.3d 1293, 1315

(Fed. Cir. 2007); *Telecordia Techs., Inc. v. Cisco Sys., Inc.*, 612 F.3d 1365, 1378–79 (Fed. Cir. 2010) ("If the district court determines that a permanent injunction is not warranted, the district court may, and is encouraged, to allow the parties to negotiate a license."). It is only when these negotiations fail and the patentee files the appropriate motion asking the Court to set an ongoing royalty rate that the Court considers whether a royalty should be imposed. *See Whitserve, LLC v. Computer Packages, Inc.*, 694 F.3d 10, 35 (Fed. Cir. 2012) (noting there are several types of relief for ongoing infringement that a court can consider but that a court may also decide that “no forward-looking relief is appropriate.”). Inherently, this procedure is time consuming.[3] Therefore, whenever the Court wishes to bring finality to the original action as quickly as possible, the Court severs the issue of ongoing infringement into a new cause of action. *See Advanced Neuromodulation Sys, Inc. v. Advanced Bionics Corp.*, 2005 U.S. Dist. LEXIS 47695, at *2 (E.D. Tex. 2005) (“A claim may be severed if it will serve the ends of justice and further the prompt and efficient disposition of litigation.”).

Additionally, severing the issues of ongoing infringement and royalty allows the original judgment to become final and proceed on its appellate course. Since determination of post-judgment issues of ongoing infringement and royalty may be time consuming and a moving target, awaiting determination of these issues would substantially delay appeal and final resolution of the case. Here, the Court exercised its discretion and severed the cause of action for ongoing infringement and the request for an ongoing royalty into a separate action. Since VirnetX’s continuing cause of action for damages due to Apple’s continuing post-judgment

[3] The Court notes that a request for an ongoing royalty goes beyond simply performing a mathematical calculation to determine future damages. Post-judgment damages are analyzed differently than pre-judgment damages. *See Amado v. Microsoft Corp.*, 517 F.3d 1353, 1361– 62 (Fed. Cir. 2008). Additionally, an ongoing royalty applies to accused products and modified versions of those products, so long as those modified versions are not “colorably different” from the adjudicated products. *Creative Internet Advertising Corp. v. Yahoo! Inc.*, 674 F. Supp. 2d 847, 854 (E.D. Tex. 2009). Therefore, the trial court must also determine the issue of whether modified versions are or are not “colorably different” from the adjudicated products when resolving a request for an ongoing royalty.

infringement of the patents-in-suit was properly severed from the original case, Apple's motion is denied.

CONCLUSION

For the foregoing reasons, the Court **DENIES** Apple's Motion to Amend or Alter the Final Judgment.

So ORDERED and SIGNED this 4th day of June, 2013.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

TAB 6

U 7377937

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

September 18, 2012

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *6,502,135*
ISSUE DATE: *December 31, 2002*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

R. Pondexter

R. PONDEXTER
Certifying Officer

UNITED STATES PATENT AND TRADEMARK OFFICE
DEPARTMENT OF COMMERCE

Plaintiffs' VirnetX Exhibit
VirnetX, Inc. v. Apple, Inc.
PX001
C.A. 6:10-cv-0417

US006502135B1

(12) United States Patent
Munger et al.

(10) Patent No.: US 6,502,135 B1
(45) Date of Patent: Dec. 31, 2002

(54) AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY

(75) Inventors: **Edmund Colby Munger**, Crownsville, MD (US); **Douglas Charles Schmidt**, Severna Park, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **Science Applications International Corporation**, San Diego, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/504,783**

(22) Filed: **Feb. 15, 2000**

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999

(60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, and provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**^{7} **G06F 15/173**

(52) **U.S. Cl.** **709/225**; 709/229; 709/245

(58) **Field of Search** 709/249, 223, 709/225, 229, 245; 713/201

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,933,846 A 6/1990 Humphrey et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

DE	0 838 930	12/1999
DE	199 24 575	12/1999
EP	2 317 792	4/1998
EP	0 858 189	8/1998
GB	0 814 589	12/1997
WO	WO 98/27783	6/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/70458	11/2000
WO	WO 01 50688	7/2001

OTHER PUBLICATIONS

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963–967.

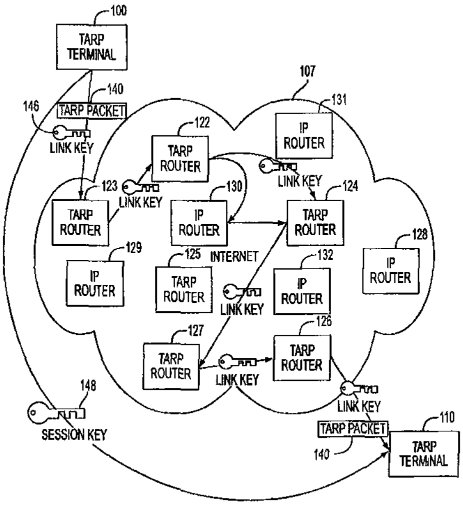
(List continued on next page.)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—Banner & Witcoff, Ltd.

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

17 Claims, 35 Drawing Sheets



U.S. PATENT DOCUMENTS

5,588,060	A		12/1996	Aziz	
5,689,566	A		11/1997	Nguyen	
5,796,942	A		8/1998	Esbensen	
5,805,801	A		9/1998	Holloway et al.	
5,842,040	A		11/1998	Hughes et al.	
5,878,231	A	*	3/1999	Baehr et al.	709/243
5,892,903	A		4/1999	Klaus	
5,898,830	A	*	4/1999	Wesinger et al.	709/225
5,905,859	A		5/1999	Holloway et al.	
6,006,259	A		12/1999	Adelman et al.	
6,016,318	A	*	1/2000	Tomoike	370/338
6,052,788	A		4/2000	Wesinger, Jr. et al.	
6,079,020	A	*	6/2000	Liu	713/201
6,119,171	A		9/2000	Alkhatib	
6,178,505	B1	*	1/2001	Schneider et al.	713/168
6,226,751	B1	*	5/2001	Arrow et al.	370/351
6,243,749	B1		6/2001	Sitaraman et al.	
6,286,047	B1	*	9/2001	Ramanathan et al.	345/733
6,330,562	B1	*	12/2001	Boden et al.	707/10
6,332,158	B1	*	12/2001	Risley et al.	709/219
6,353,614	B1	*	3/2002	Borella et al.	370/389

OTHER PUBLICATIONS

Linux FreeS/WAN Index File, printed from http://liberty-.freeswan.org/freeswan_trees/freeswan-1.3/doc/ on Feb. 21, 2002, 3 pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_trees/freesswan-1.3/doc/rationale.html on Feb. 21, 2002, 4 pages.

Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan/org/freeswan_trees/freeswan-1.3/doc/glossary.html on Feb. 21, 2002, 25 pages.

Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ss13/draft302.txt on Feb. 4, 2002, 56 pages.

Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), "Crowds: Anonymity for Web Transactions", pp. 1–23.

Dolev, Shlomi and Ostrovsky, Rafail, "Efficient Anonymous Multicast and Reception" (Extended Abstract), 16 pages.

Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82–94.

Shree Murthy et al., "Congestion–Oriented Shortest Multipath Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028–1036.

Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1–14.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group, Apr. 1998, 51 pages.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278–297 and pp. 351–375.

P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.

Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.

W. Stallings, "Cryptography And Network Security", 2^{nd} Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399–400.

W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.

* cited by examiner

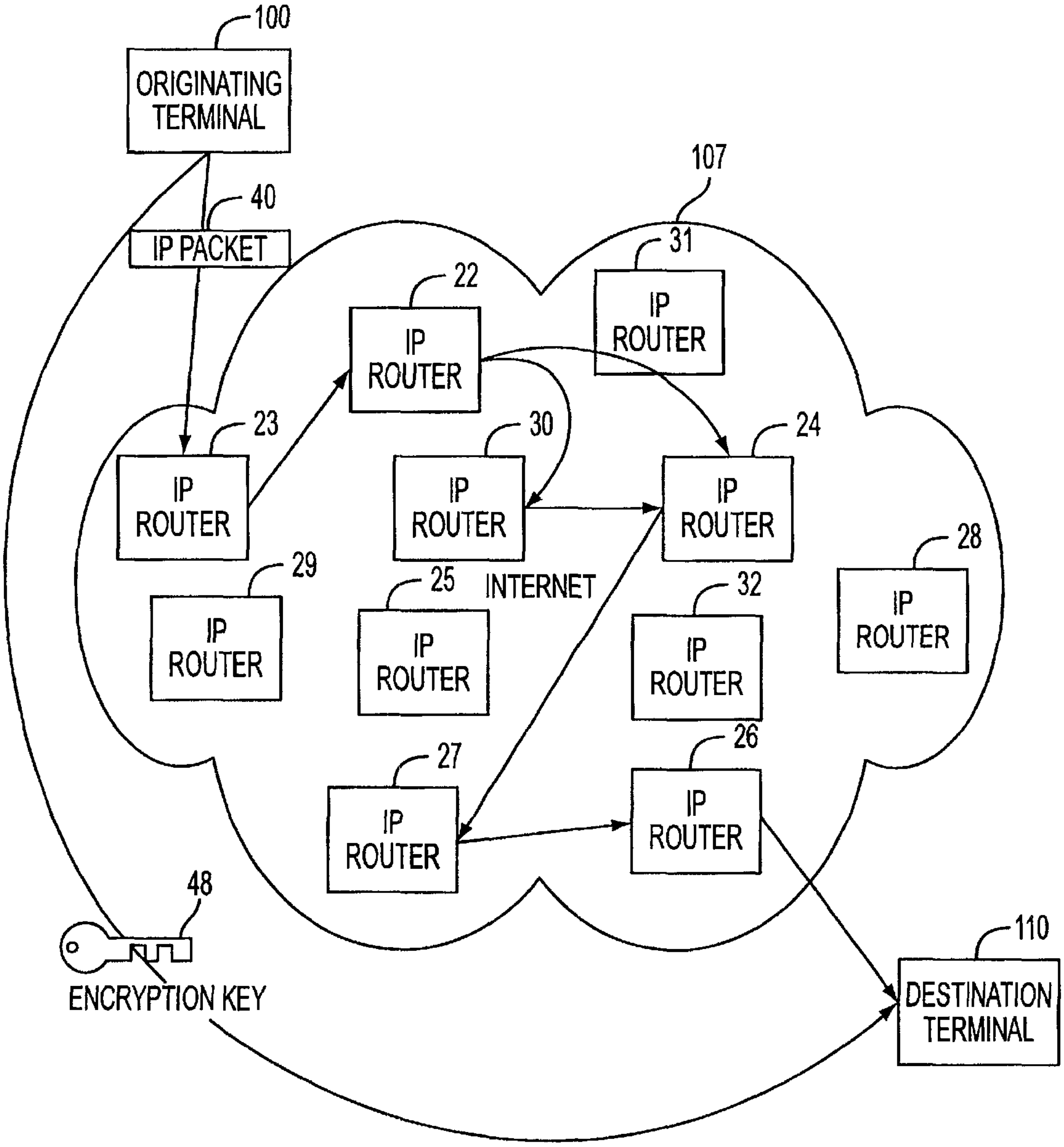

100
ORIGINATING TERMINAL
40
IP PACKET
107
31
IP ROUTER
22
IP ROUTER
23
IP ROUTER
30
IP ROUTER
24
IP ROUTER
29
IP ROUTER
25
IP ROUTER
INTERNET
32
IP ROUTER
28
IP ROUTER
27
IP ROUTER
26
IP ROUTER
48
ENCRYPTION KEY
110
DESTINATION TERMINAL

FIG. 1

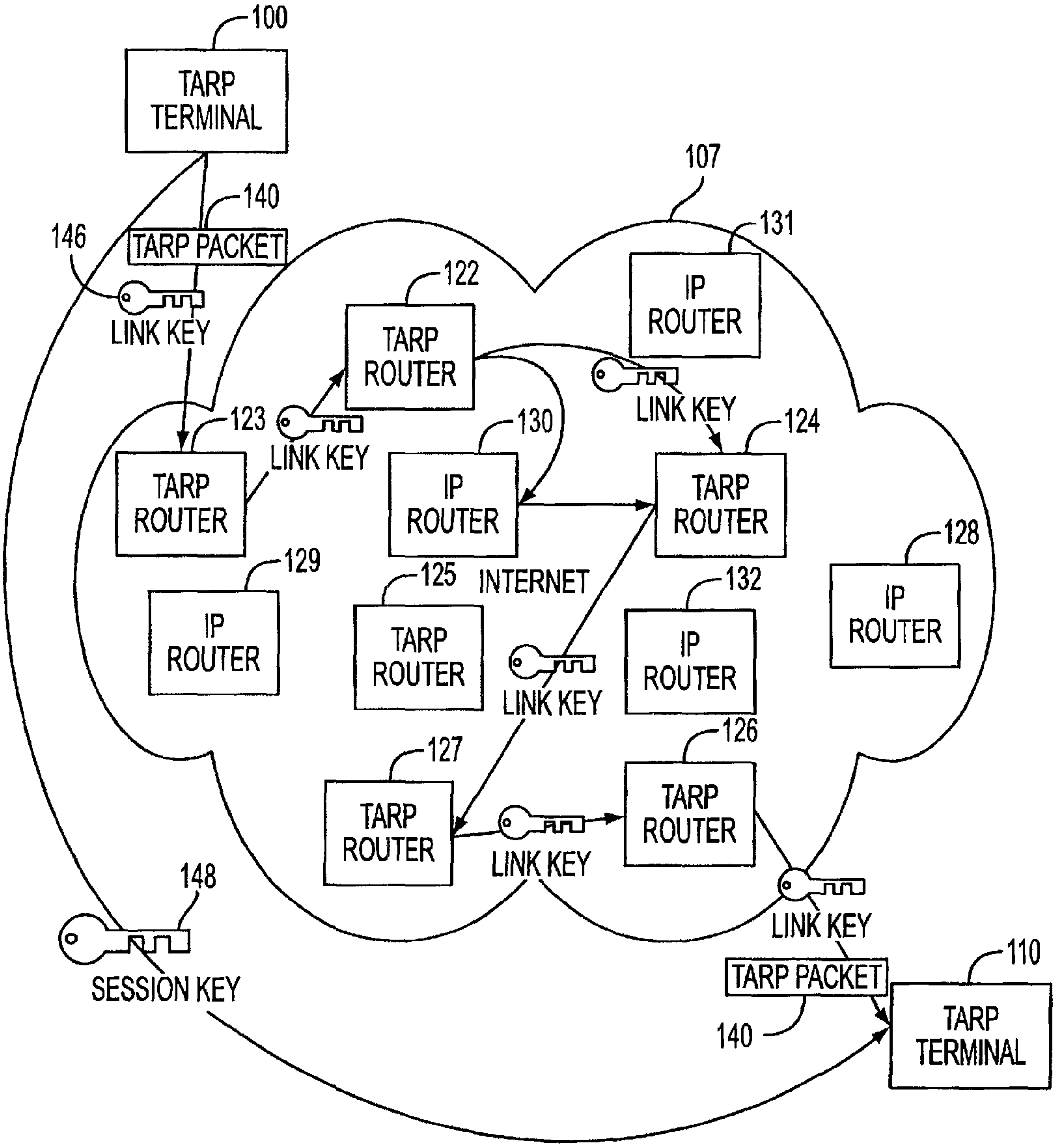

100
TARP TERMINAL
107
140
TARP PACKET
146
LINK KEY
131
IP ROUTER
122
TARP ROUTER
LINK KEY
123
LINK KEY
130
124
TARP ROUTER
IP ROUTER
TARP ROUTER
128
IP ROUTER
129
IP ROUTER
125
INTERNET
132
TARP ROUTER
IP ROUTER
LINK KEY
127
126
TARP ROUTER
TARP ROUTER
LINK KEY
148
LINK KEY
SESSION KEY
TARP PACKET
110
140
TARP TERMINAL

FIG. 2

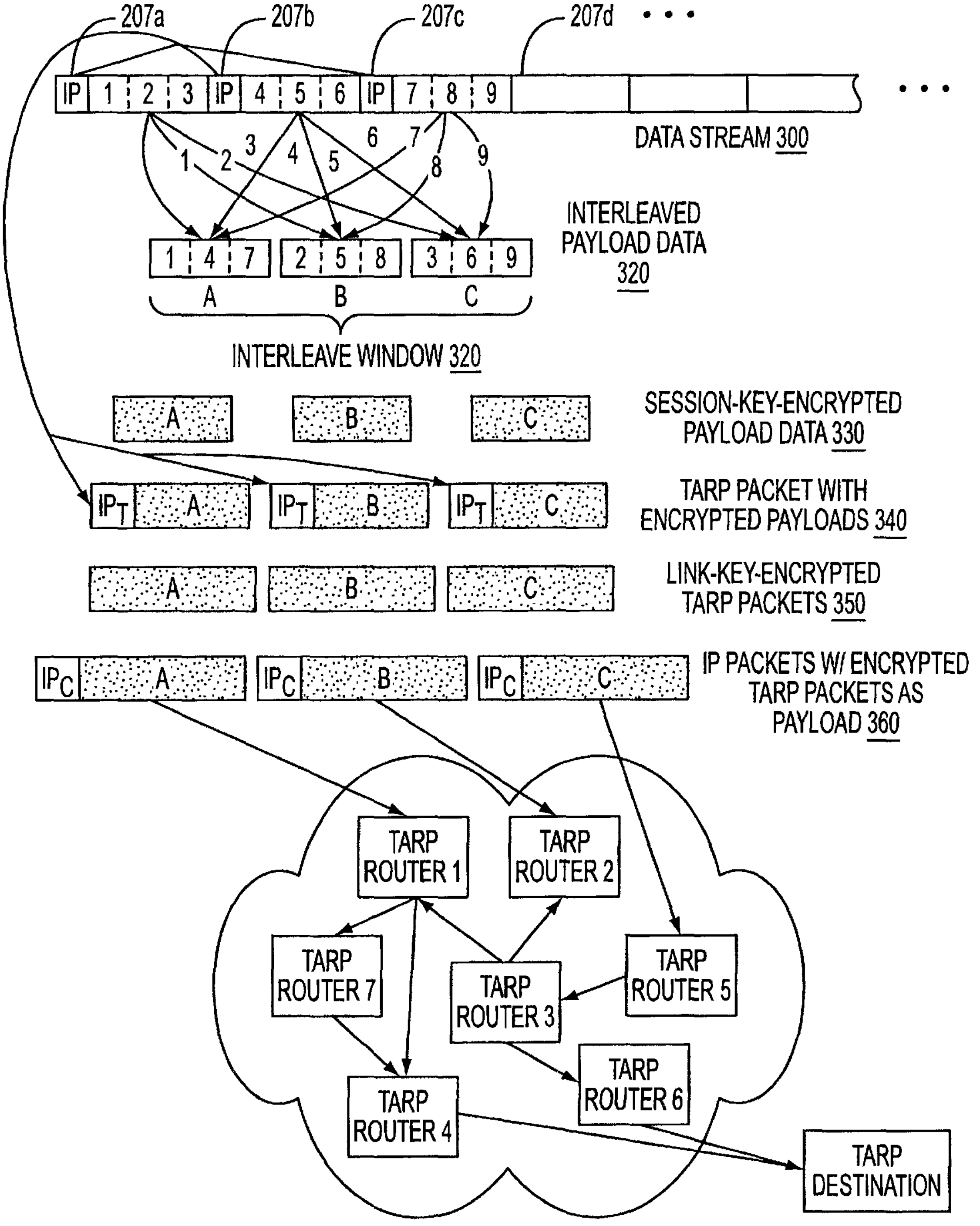

207a
207b
207c
207d
IP
1
2
3
IP
4
5
6
IP
7
8
9
DATA STREAM 300
INTERLEAVED PAYLOAD DATA 320
1 4 7
2 5 8
3 6 9
A
B
C
INTERLEAVE WINDOW 320
SESSION-KEY-ENCRYPTED PAYLOAD DATA 330
IP_T A
IP_T B
IP_T C
TARP PACKET WITH ENCRYPTED PAYLOADS 340
LINK-KEY-ENCRYPTED TARP PACKETS 350
IP_C A
IP_C B
IP_C C
IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD 360
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

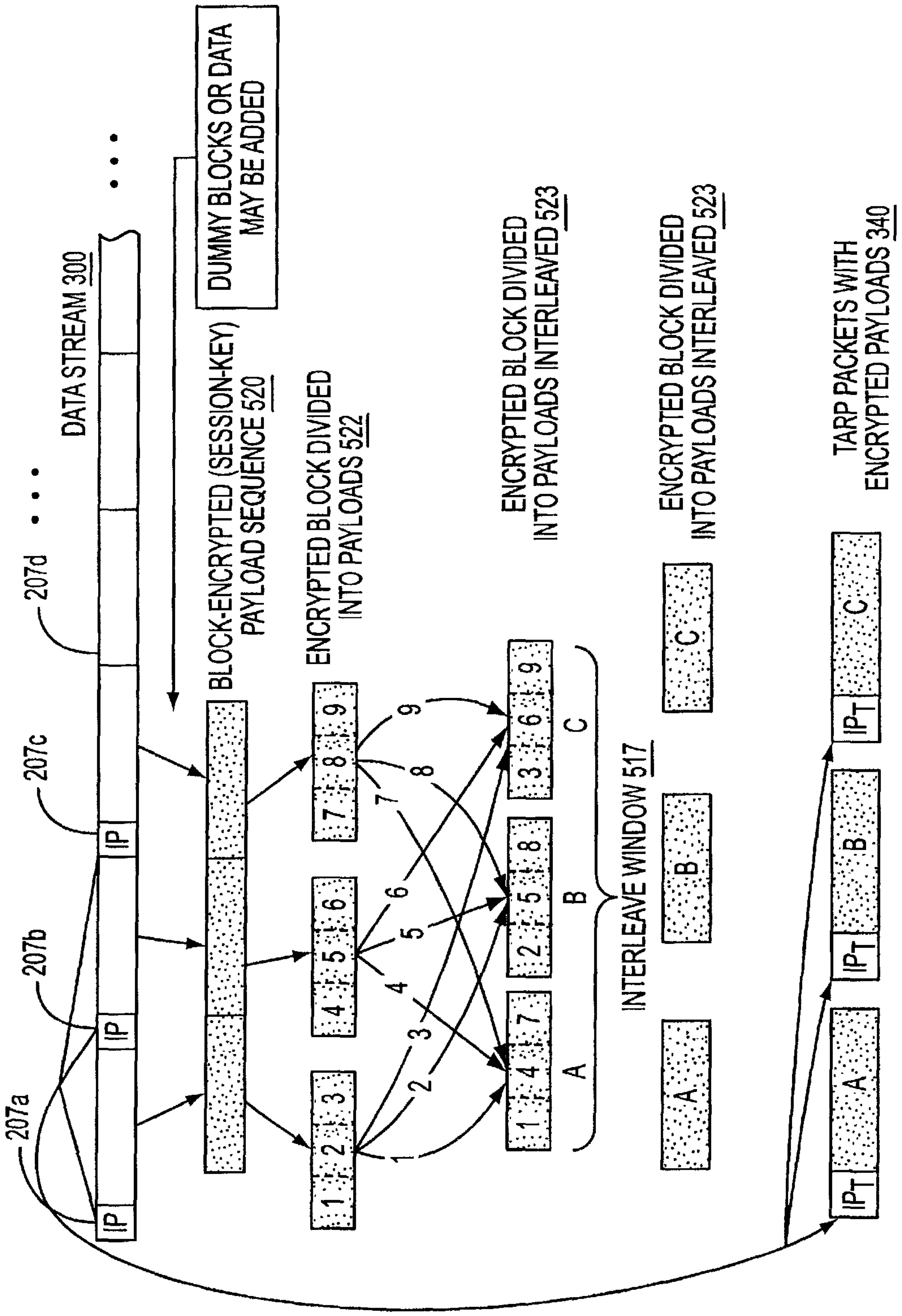

DATA STREAM 300
207a
207b
207c
207d
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE 520
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS 522
1 2 3
4 5 6
7 8 9
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
1 4 7
2 5 8
3 6 9
A
B
C
INTERLEAVE WINDOW 517
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
A
B
C
TARP PACKETS WITH ENCRYPTED PAYLOADS 340
IPT
IPT
IPT

FIG. 3B

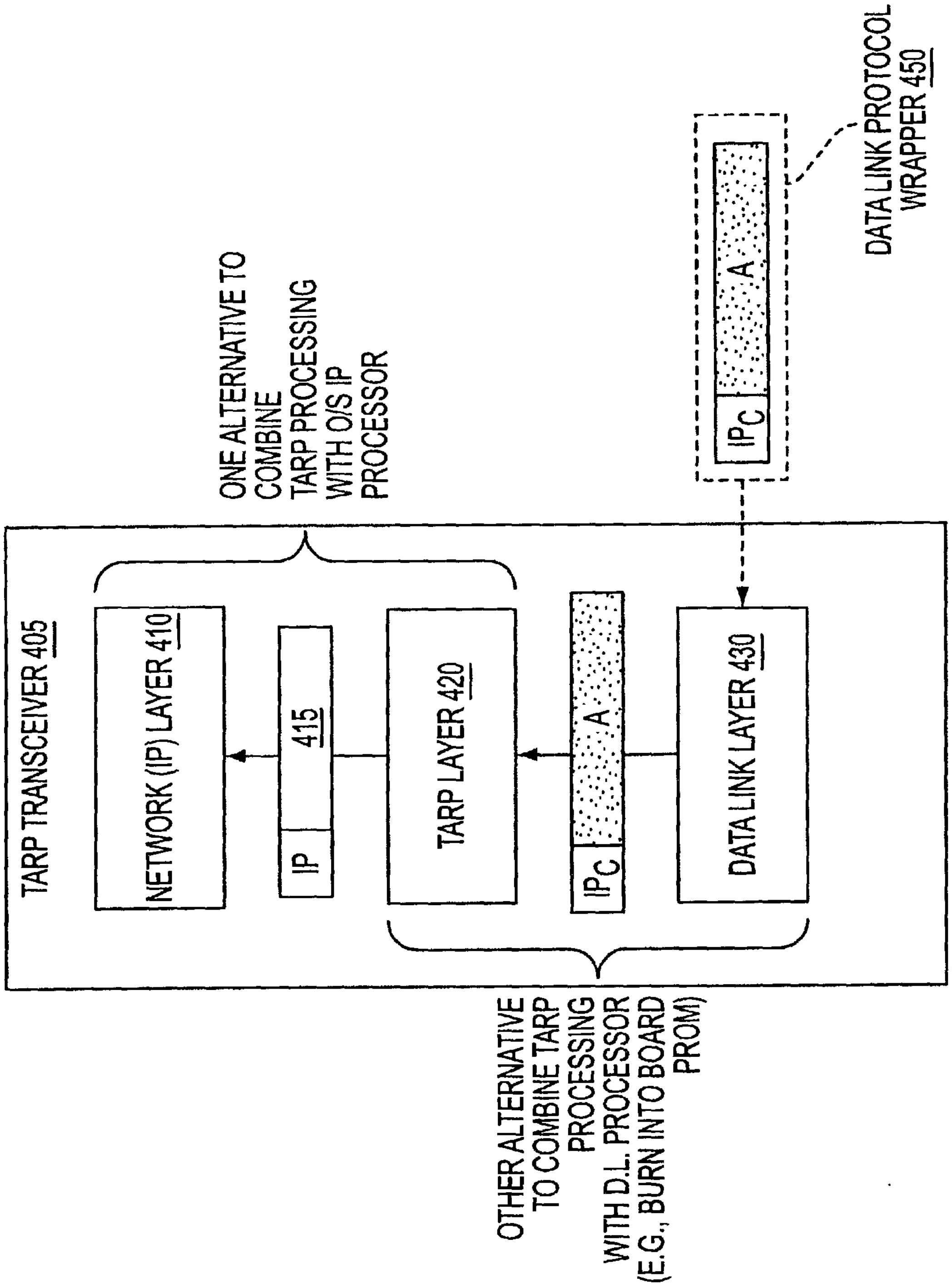

TARP TRANSCEIVER 405
NETWORK (IP) LAYER 410
IP
415
TARP LAYER 420
IPC
A
DATA LINK LAYER 430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (E.G., BURN INTO BOARD PROM)
IPC
A
DATA LINK PROTOCOL WRAPPER 450

FIG. 4

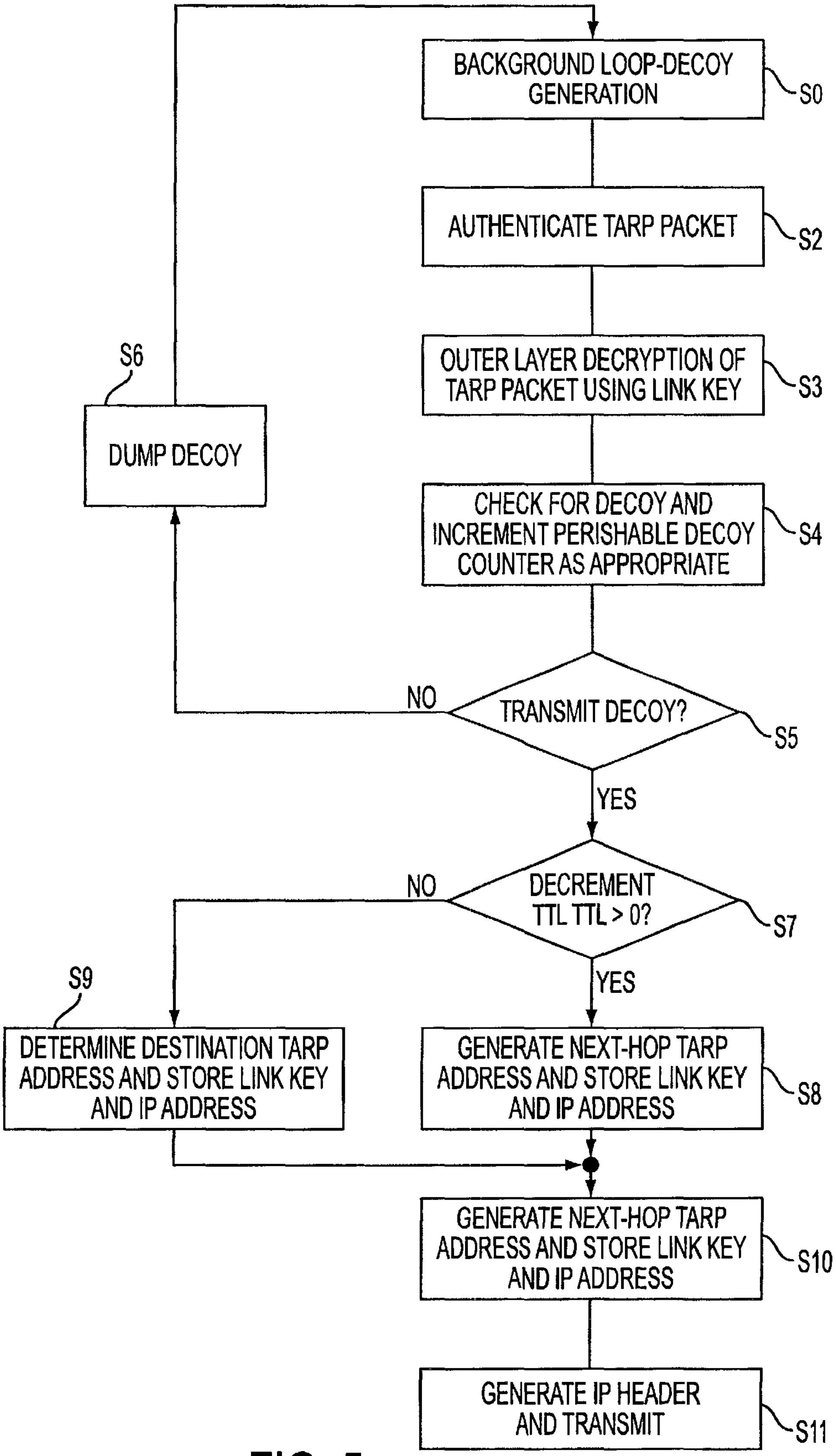

BACKGROUND LOOP-DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
S6
DUMP DECOY
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

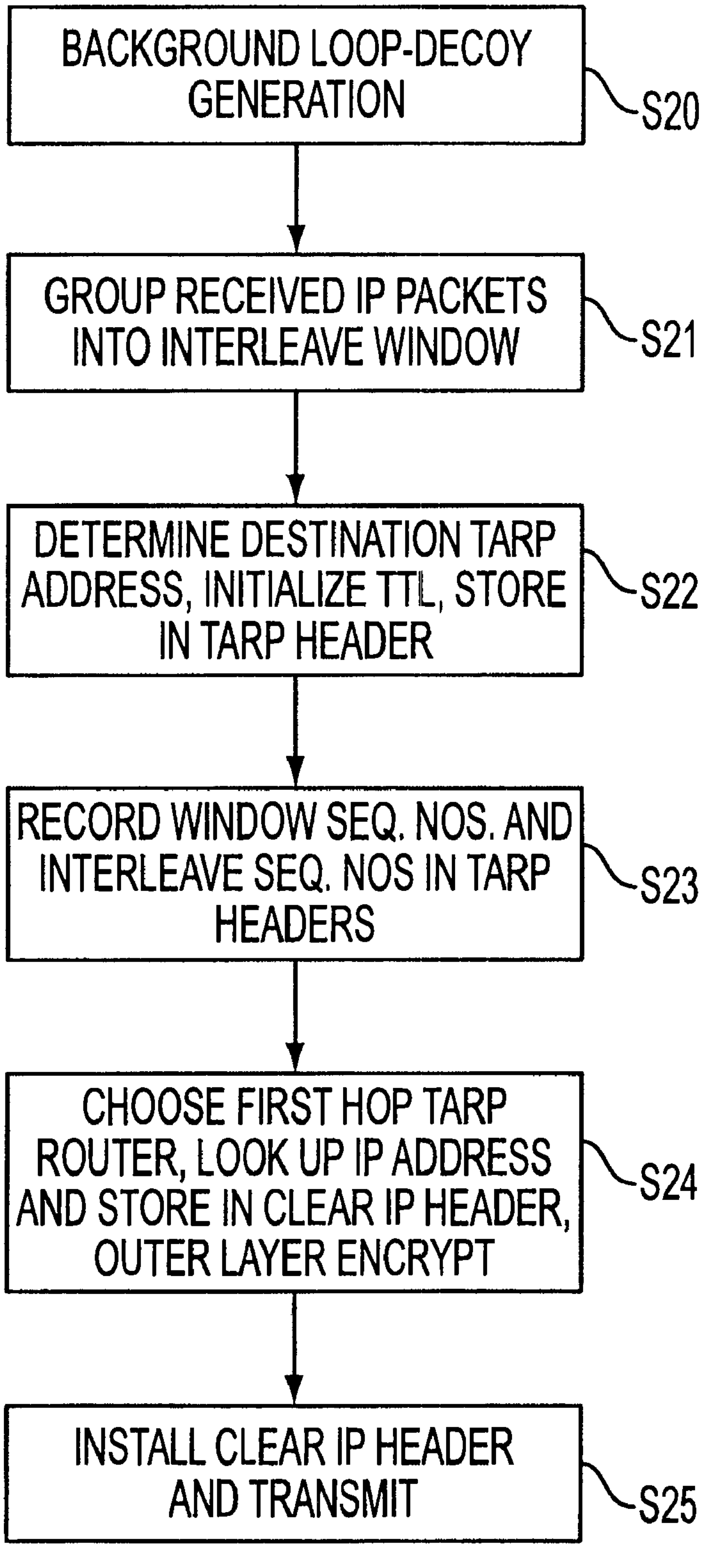

BACKGROUND LOOP-DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

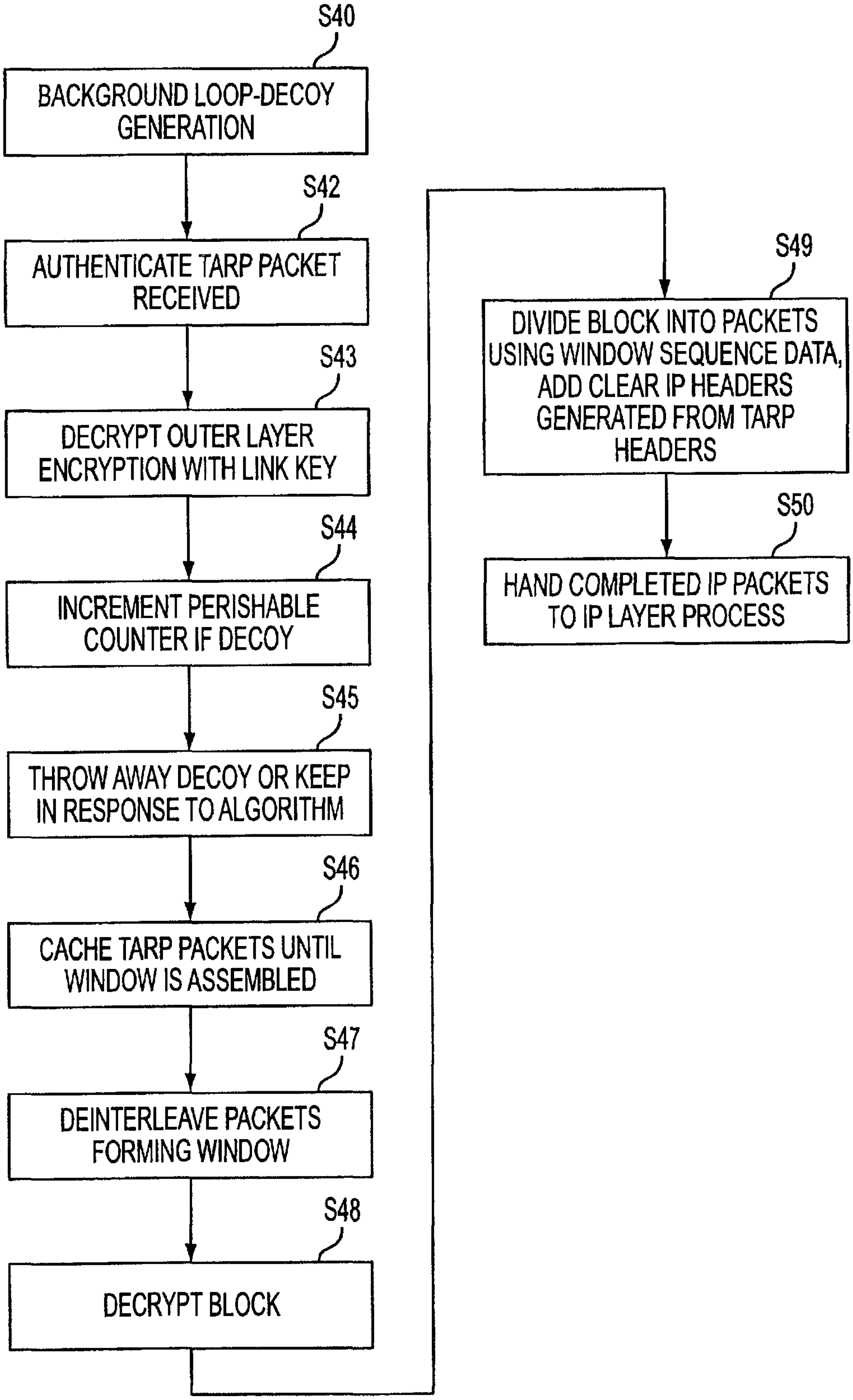

S40
BACKGROUND LOOP-DECOY GENERATION
S42
AUTHENTICATE TARP PACKET RECEIVED
S43
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S44
INCREMENT PERISHABLE COUNTER IF DECOY
S45
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S46
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S47
DEINTERLEAVE PACKETS FORMING WINDOW
S48
DECRYPT BLOCK
S49
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S50
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS

FIG. 7

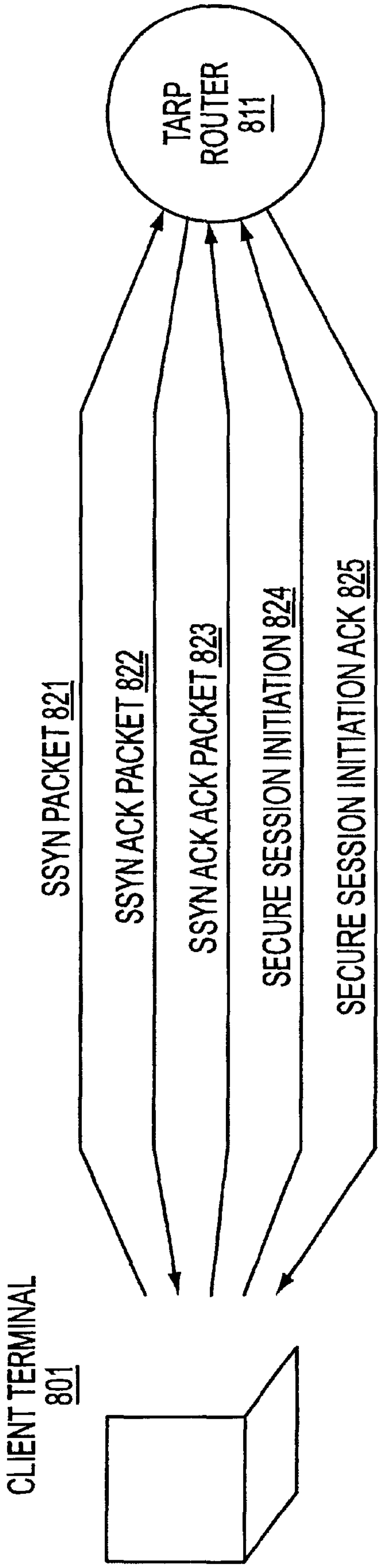
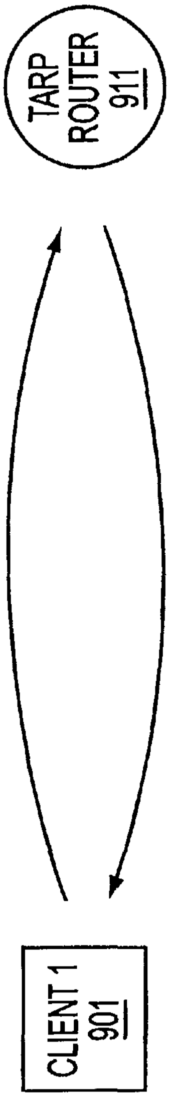

TARP
ROUTER
811
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
SECURE SESSION INITIATION 824
SECURE SESSION INITIATION ACK 825
CLIENT TERMINAL
801

FIG. 8



TRANSMIT TABLE 921

131.218.204.98 ,	131.218.204.65
131.218.204.221 ,	131.218.204.97
131.218.204.139 ,	131.218.204.186
131.218.204.12 ,	131.218.204.55
. . .	. . .

RECEIVE TABLE 922

131.218.204.161 ,	131.218.204.89
131.218.204.66 ,	131.218.204.212
131.218.204.201 ,	131.218.204.127
131.218.204.119 ,	131.218.204.49
. . .	. . .

RECEIVE TABLE 924

131.218.204.98 ,	131.218.204.65
131.218.204.221 ,	131.218.204.97
131.218.204.139 ,	131.218.204.186
131.218.204.12 ,	131.218.204.55
. . .	. . .

TRANSMIT TABLE 923

131.218.204.161 ,	131.218.204.89
131.218.204.66 ,	131.218.204.212
131.218.204.201 ,	131.218.204.127
131.218.204.119 ,	131.218.204.49
. . .	. . .

FIG. 9

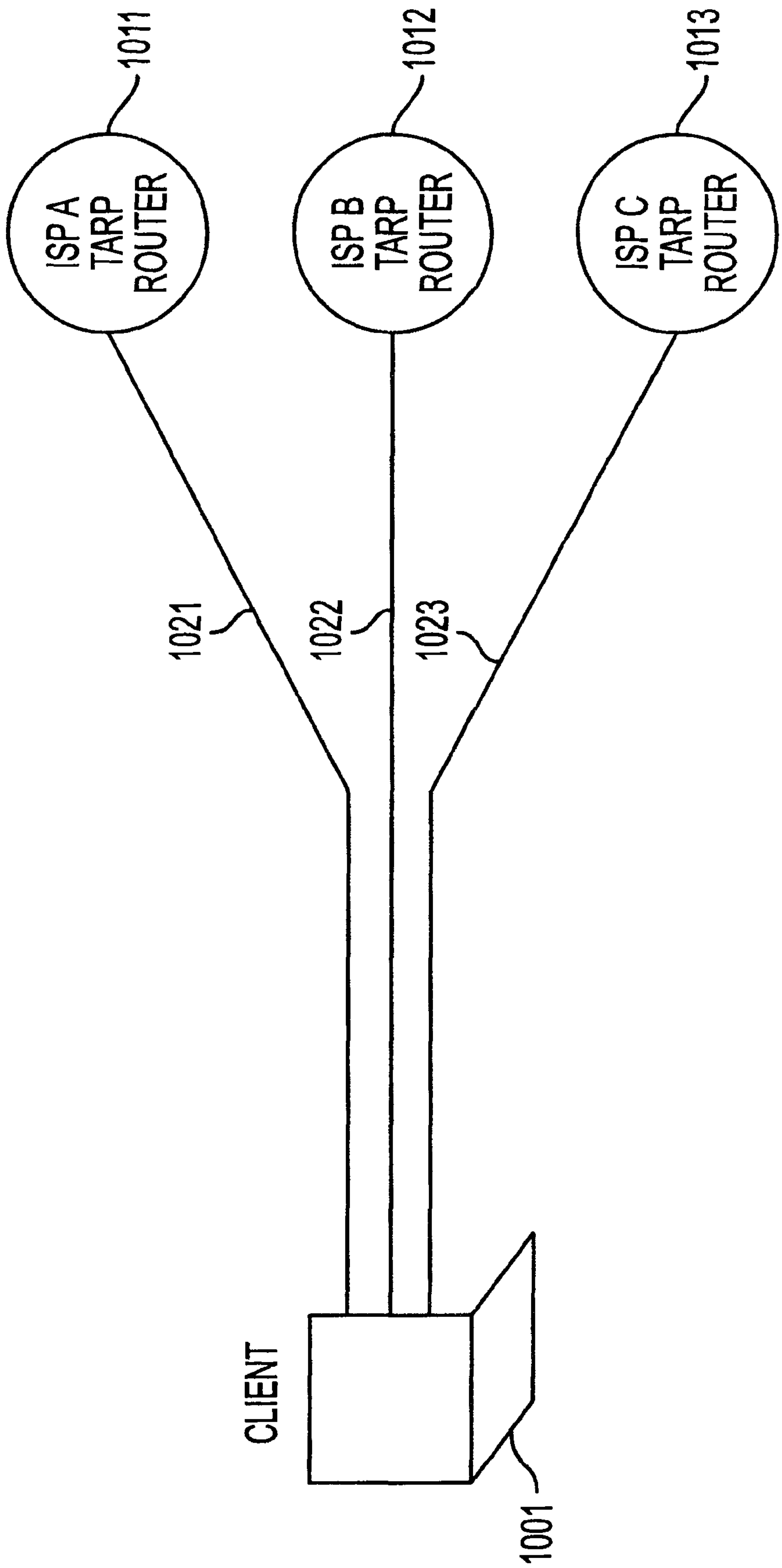

1011
1012
1013
ISP A TARP ROUTER
ISP B TARP ROUTER
ISP C TARP ROUTER
1021
1022
1023
CLIENT
1001

FIG. 10

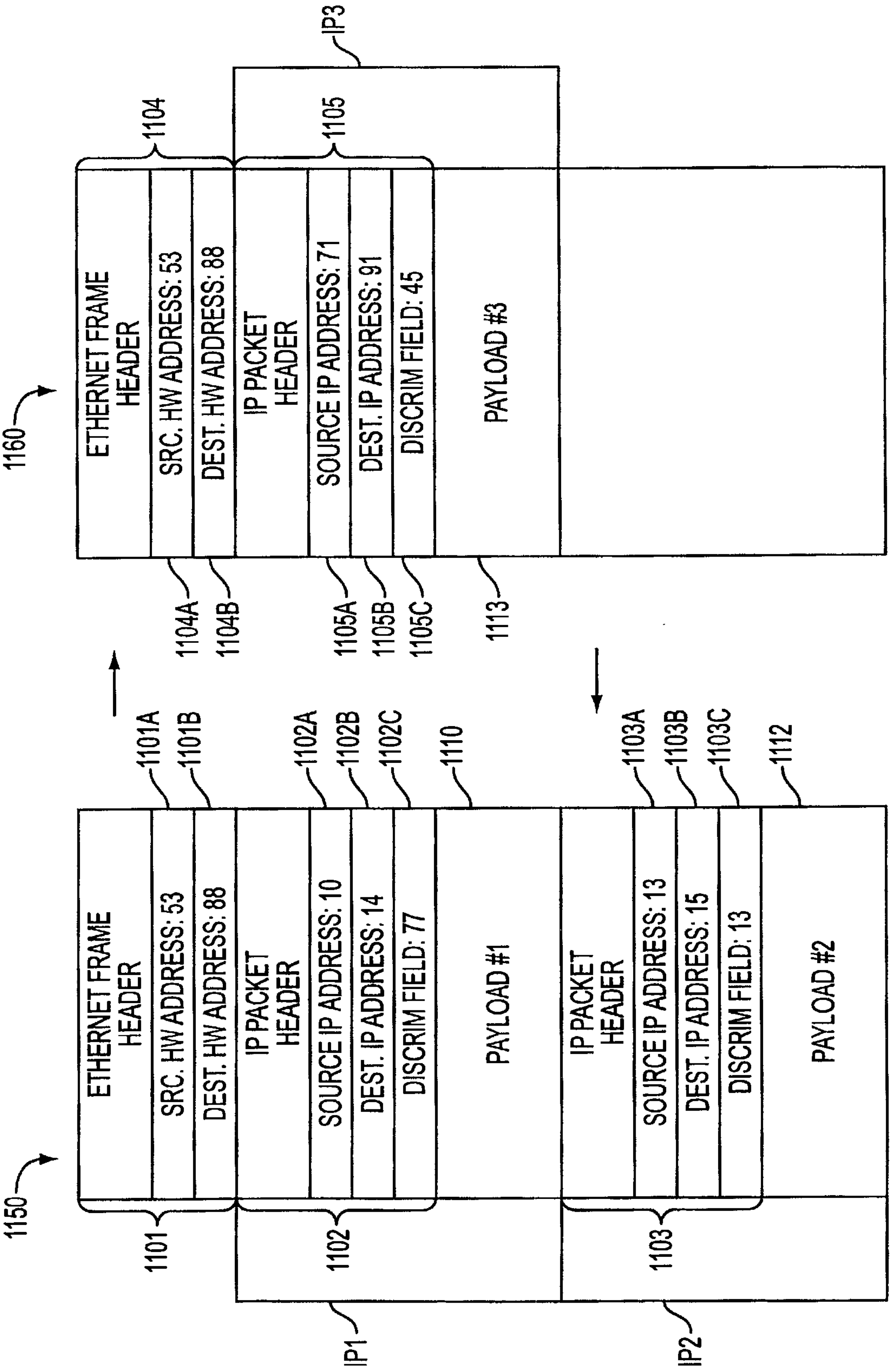
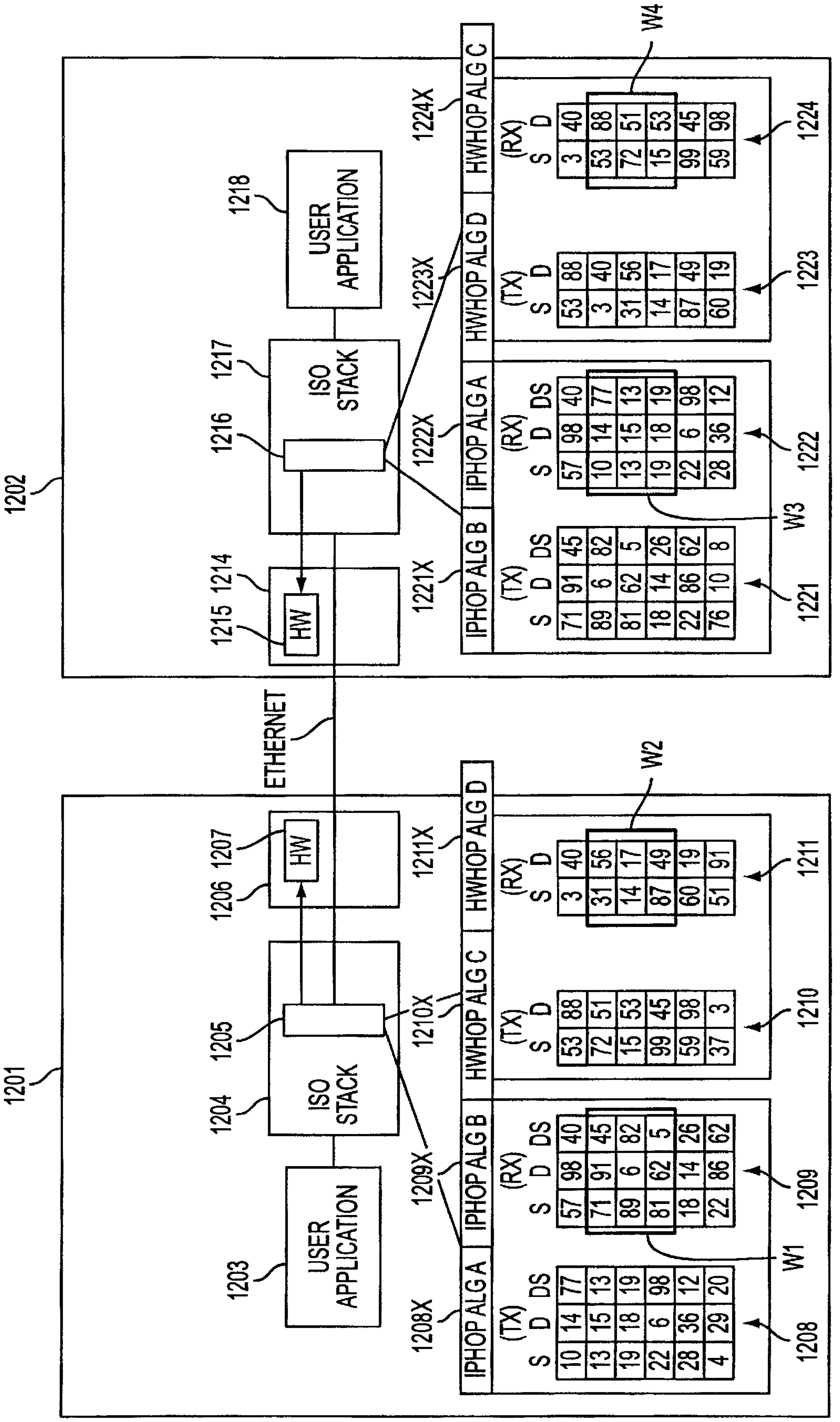

1150
1101
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
1101A
1101B
1102
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
1102A
1102B
1102C
PAYLOAD #1
1110
IP1
1103
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
1103A
1103B
1103C
PAYLOAD #2
1112
IP2
1160
1104
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
1104A
1104B
1105
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
1105A
1105B
1105C
PAYLOAD #3
1113
IP3

FIG. 11



(TX)

S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

(RX)

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

(TX)

S	D
53	88
72	51
15	53
99	45
59	98
37	3

(RX)

S	D
3	40
31	56
14	17
87	49
60	19
51	91

(TX)

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

(RX)

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

(TX)

S	D
53	88
3	40
31	56
14	17
87	49
60	19

(RX)

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

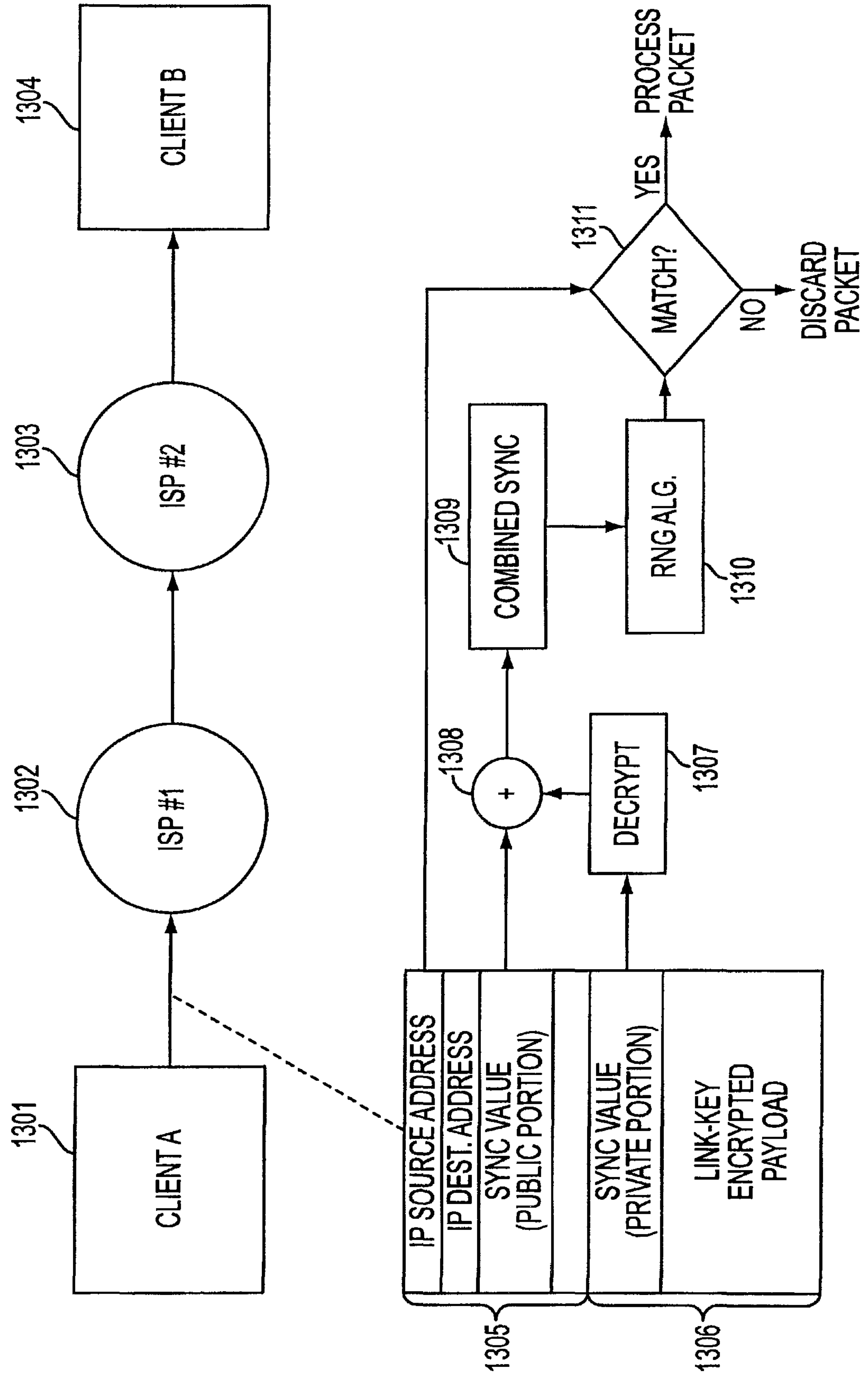

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
+
1308
COMBINED SYNC
1309
RNG ALG.
1310
MATCH?
1311
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

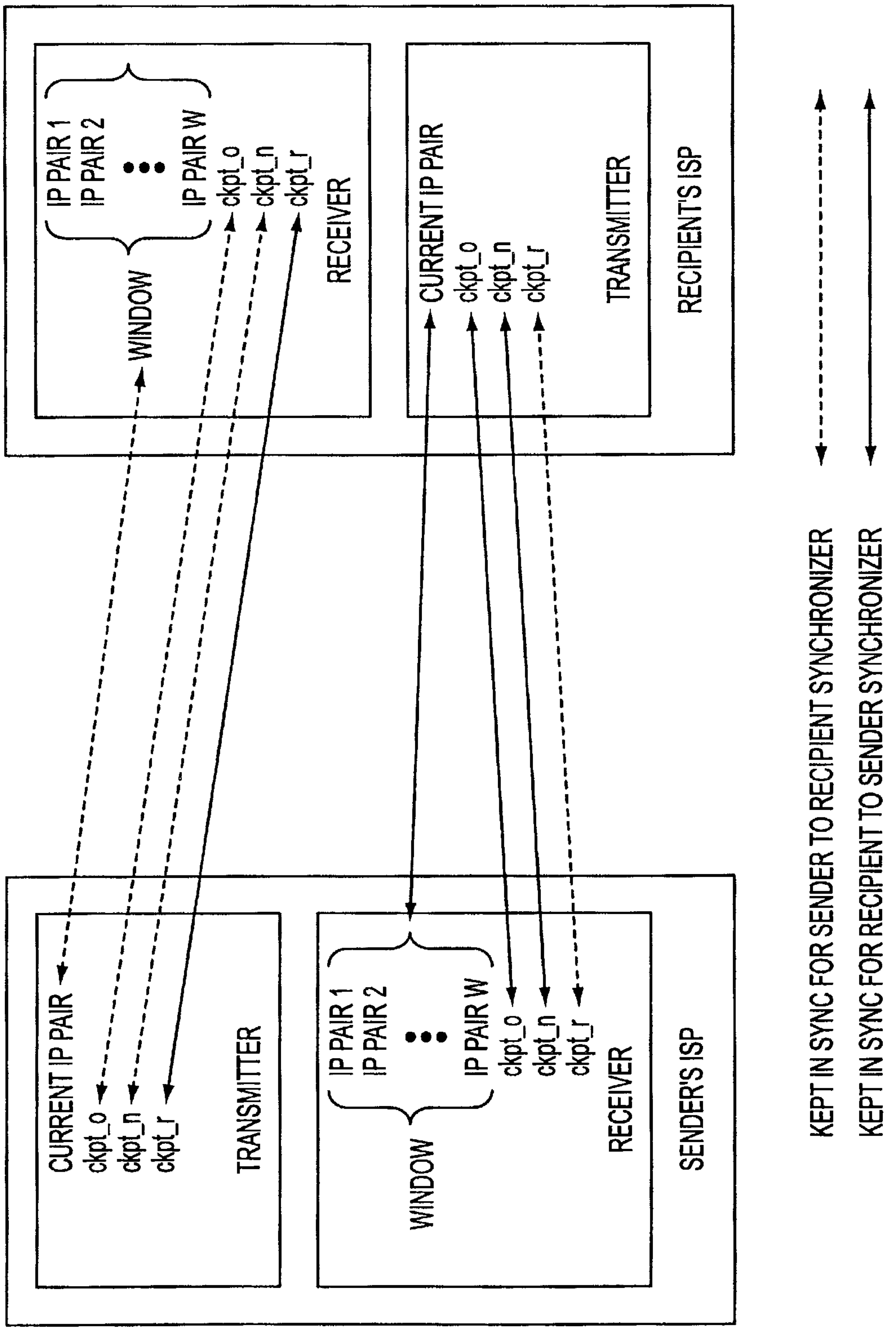
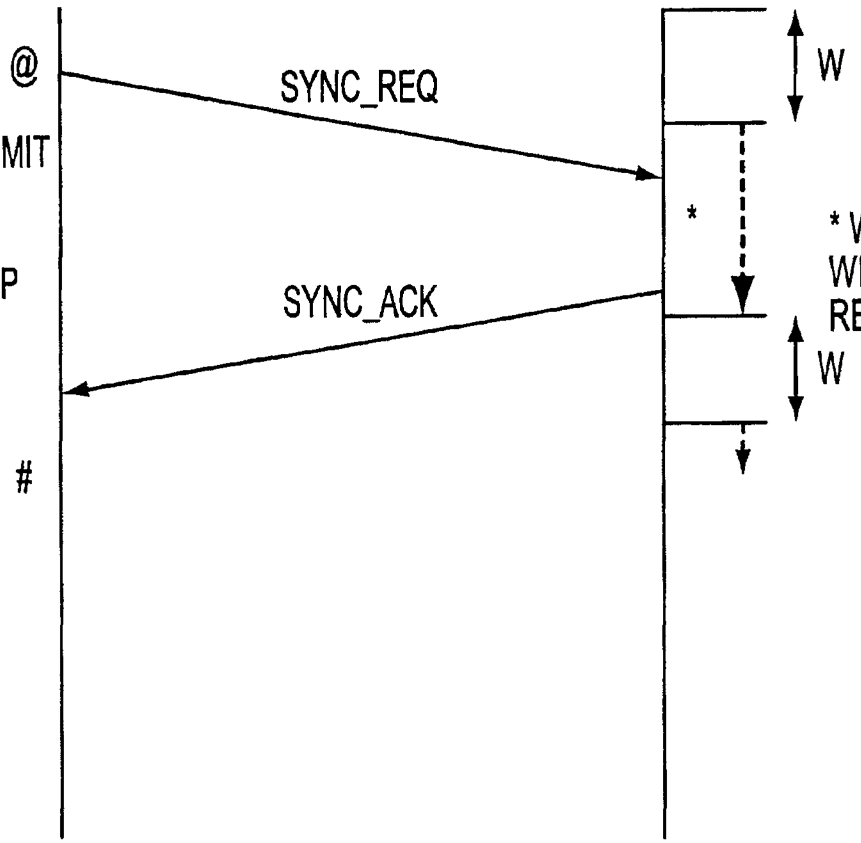

CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
WINDOW
IP PAIR 1
IP PAIR 2
•••
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
SENDER'S ISP
WINDOW
IP PAIR 1
IP PAIR 2
•••
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
RECIPIENT'S ISP
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14



@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r

WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:

- UPDATE WINDOW
- GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
- GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
- TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r

FIG. 15

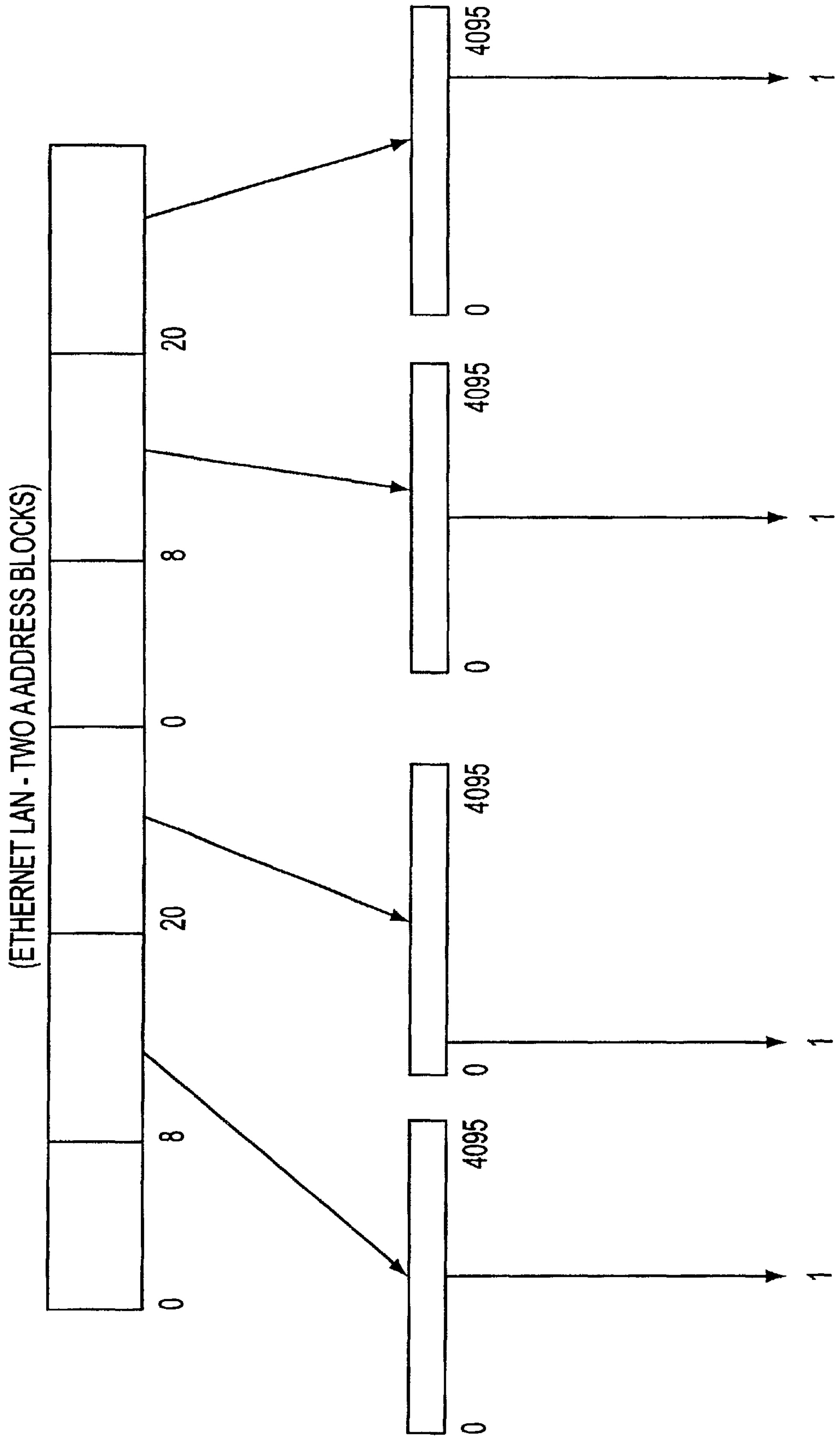

(ETHERNET LAN - TWO A ADDRESS BLOCKS)
0
8
20
0
8
20
0
4095
0
4095
0
4095
0
4095
1
1
1
1

FIG. 16

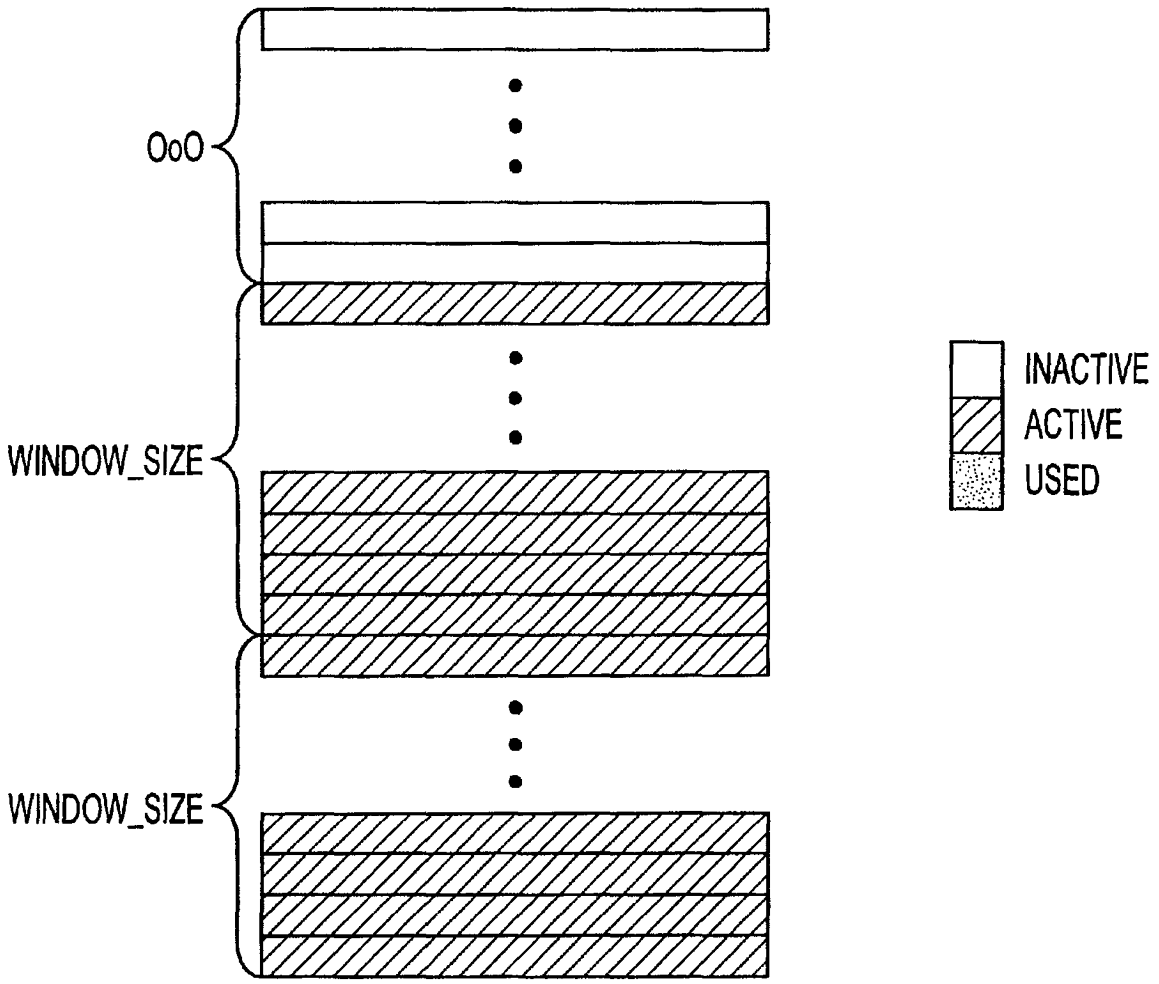


FIG. 17

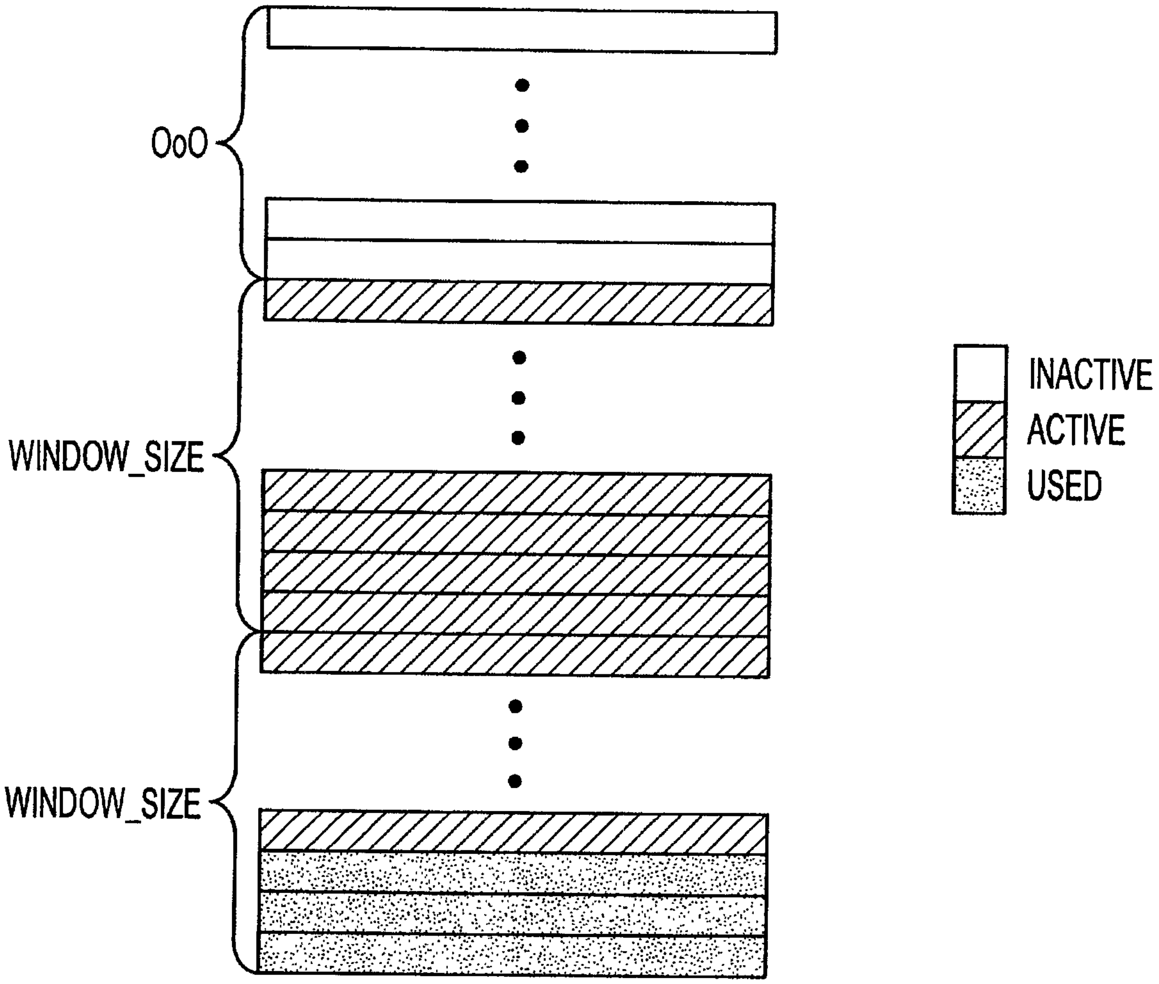


FIG. 18

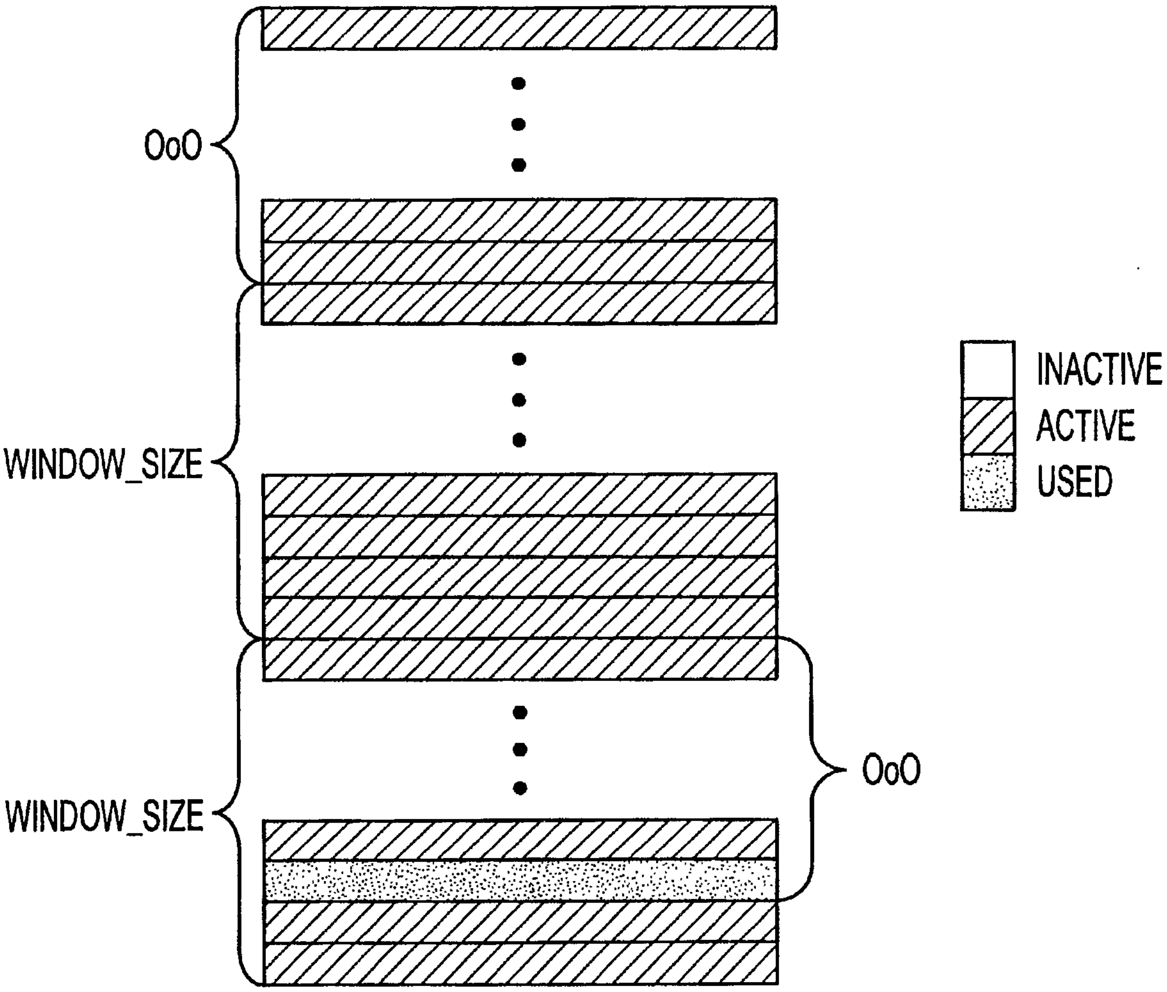


FIG. 19

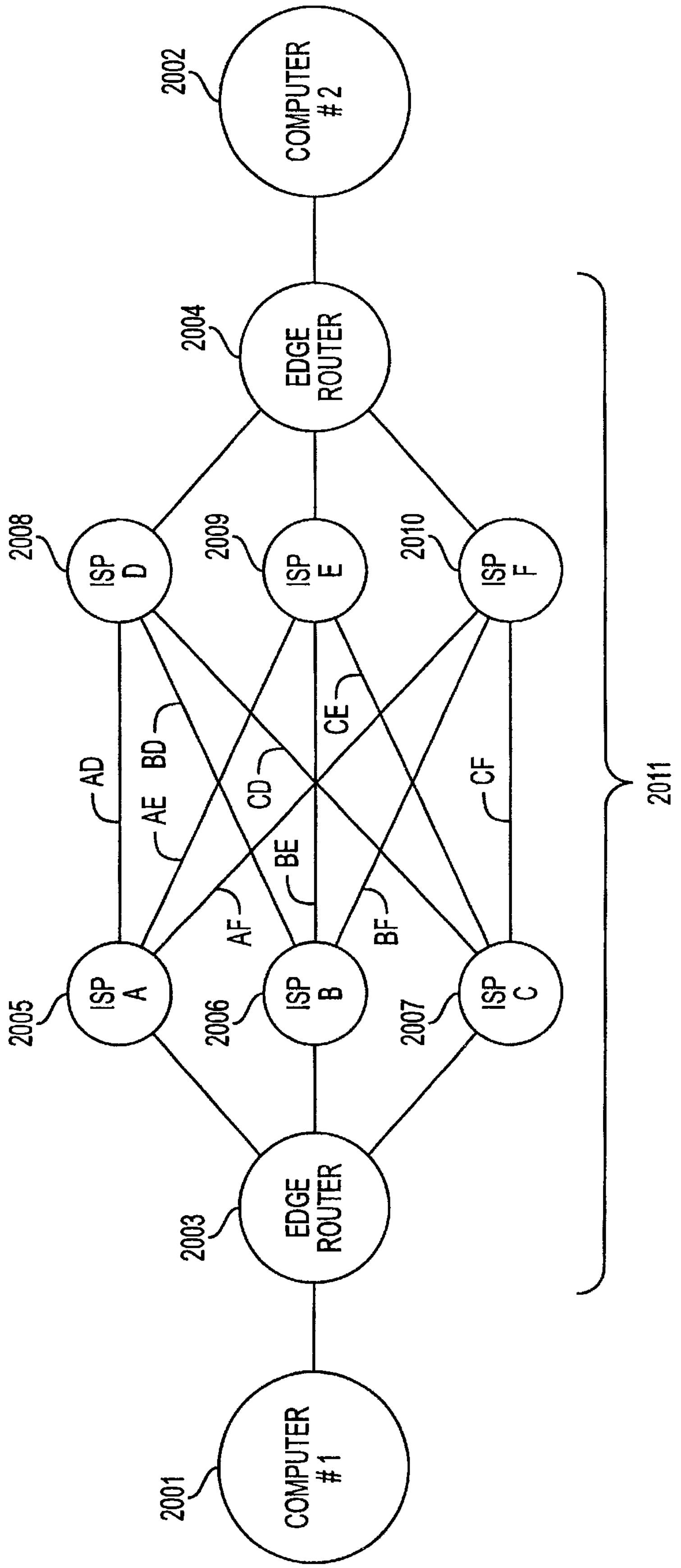

COMPUTER #2
2002
EDGE ROUTER
2004
ISP D
2008
ISP E
2009
ISP F
2010
AD
AE
BD
CD
CE
BE
AF
BF
CF
ISP A
2005
ISP B
2006
ISP C
2007
EDGE ROUTER
2003
COMPUTER #1
2001
2011

FIG. 20

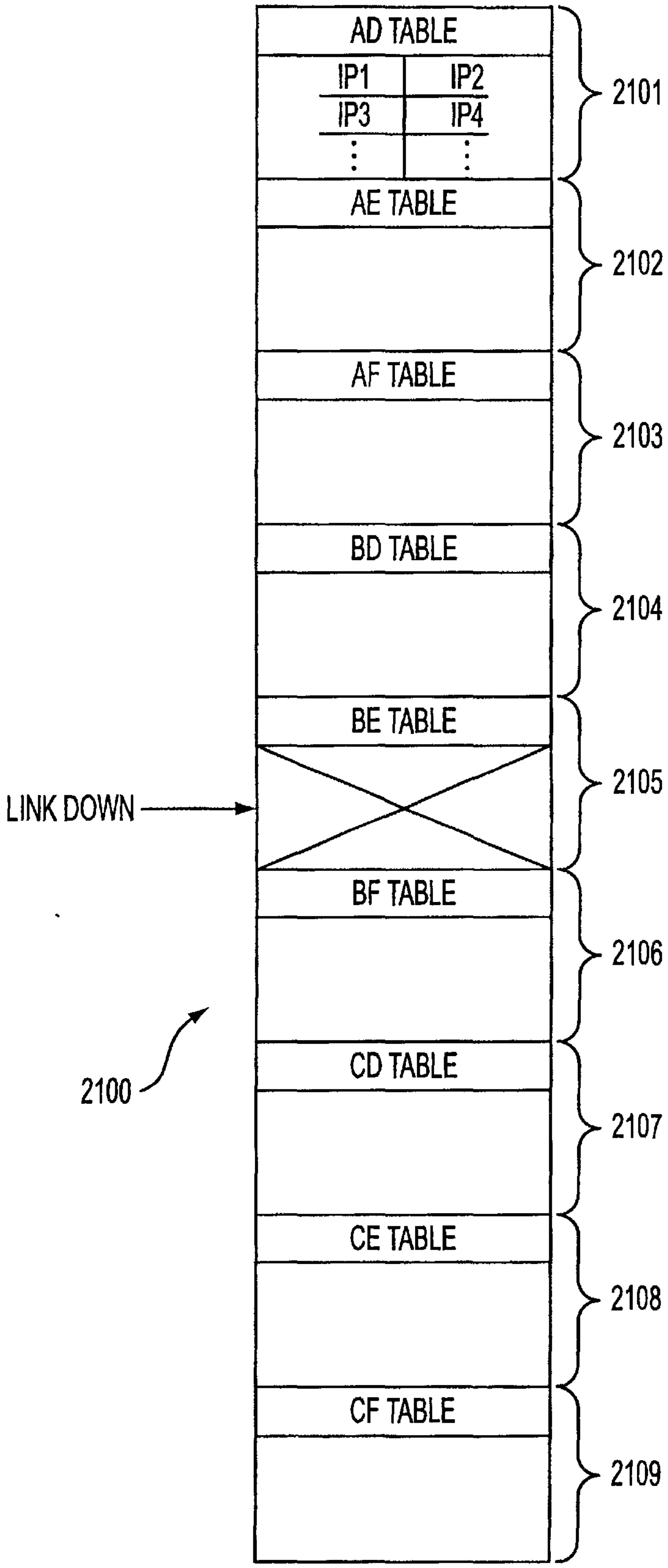

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

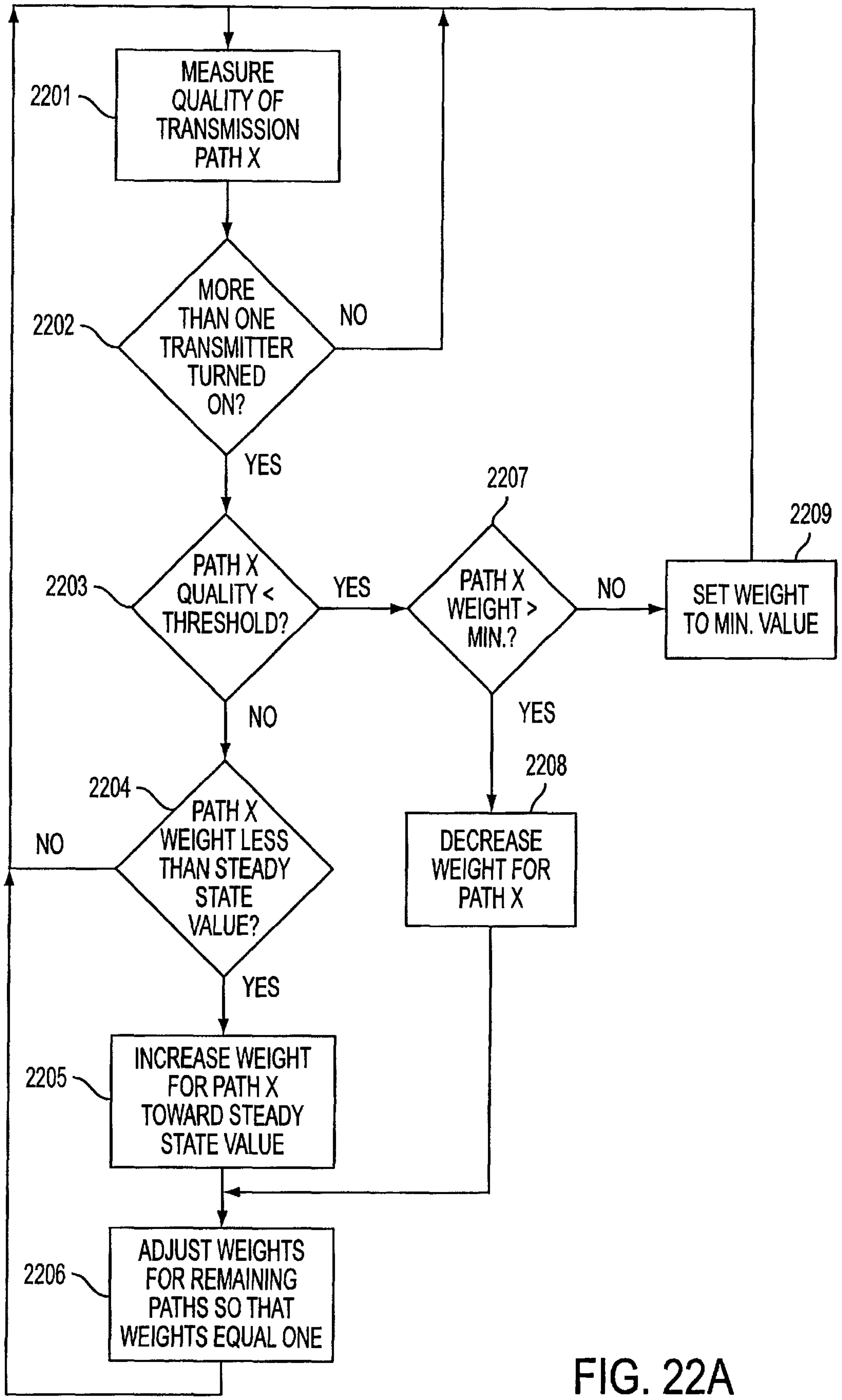

2201
MEASURE QUALITY OF TRANSMISSION PATH X
2202
MORE THAN ONE TRANSMITTER TURNED ON?
NO
YES
2203
PATH X QUALITY < THRESHOLD?
YES
NO
2207
PATH X WEIGHT > MIN.?
NO
YES
2209
SET WEIGHT TO MIN. VALUE
2204
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
YES
2208
DECREASE WEIGHT FOR PATH X
2205
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2206
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE

FIG. 22A

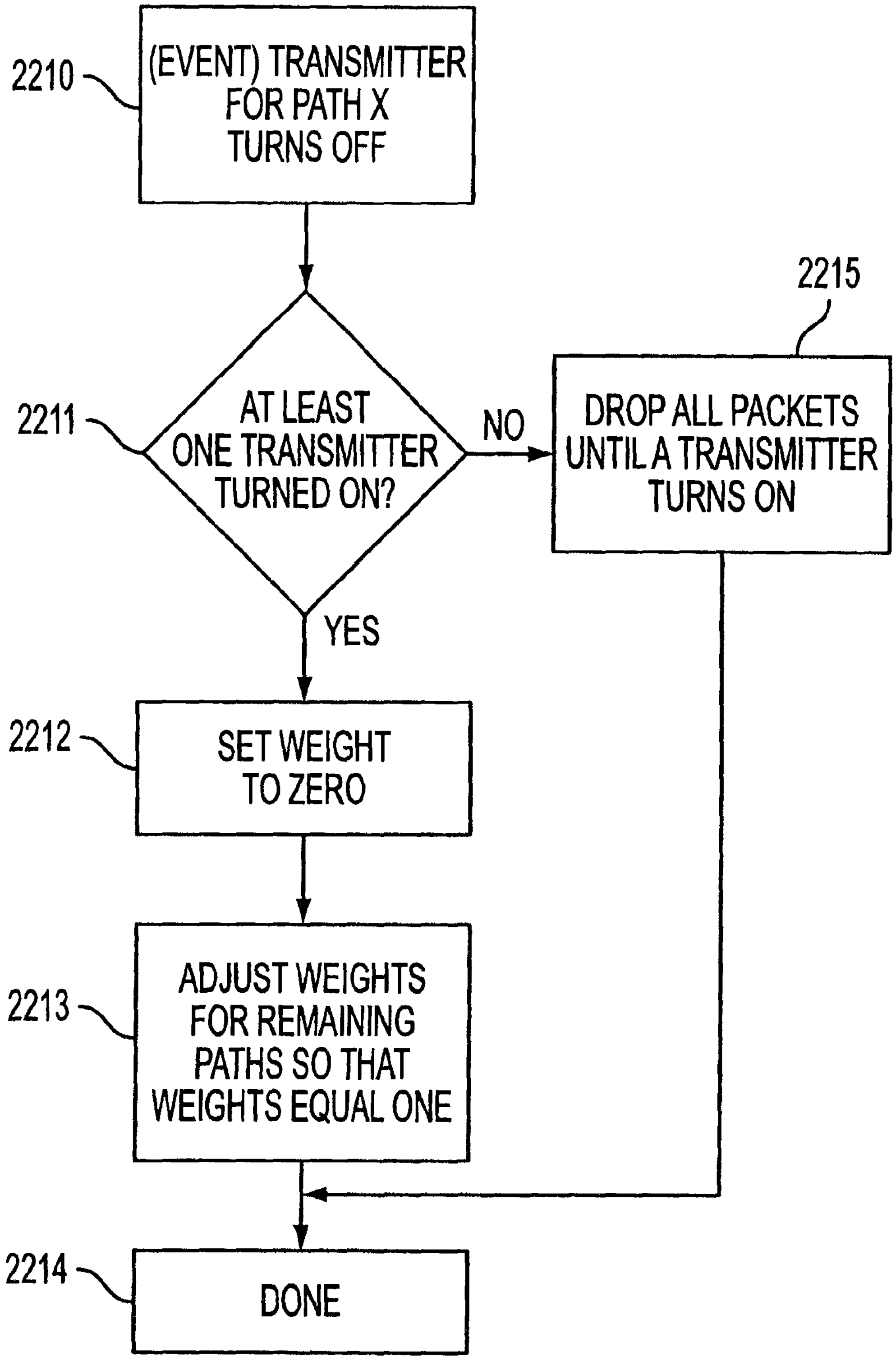

2210
(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2211
AT LEAST ONE TRANSMITTER TURNED ON?
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
2212
SET WEIGHT TO ZERO
2213
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2214
DONE

FIG. 22B

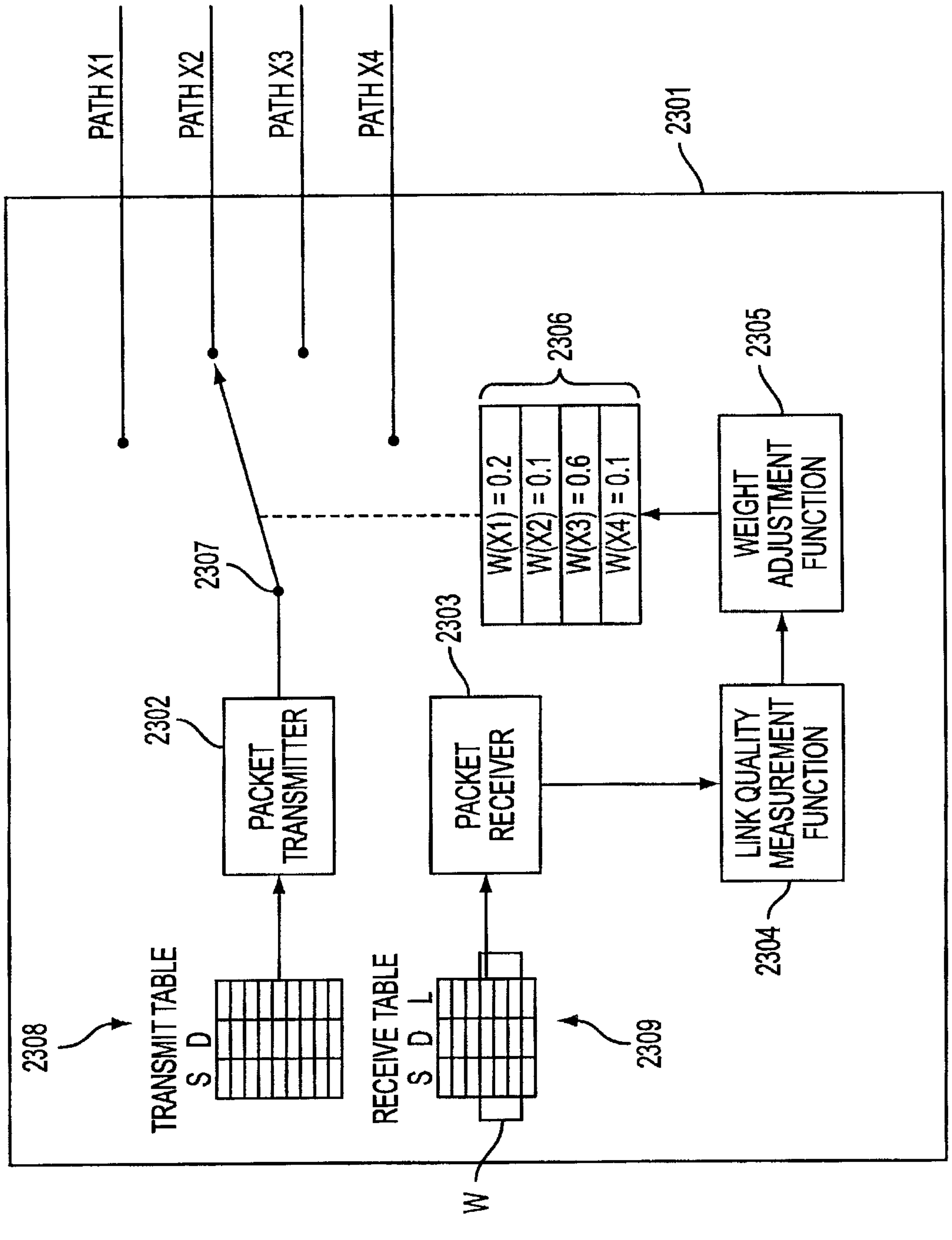

PATH X1
PATH X2
PATH X3
PATH X4
2301
2306
W(X1) = 0.2
W(X2) = 0.1
W(X3) = 0.6
W(X4) = 0.1
2305
WEIGHT ADJUSTMENT FUNCTION
2307
2303
2302
PACKET TRANSMITTER
PACKET RECEIVER
LINK QUALITY MEASUREMENT FUNCTION
2304
2308
TRANSMIT TABLE
S D
RECEIVE TABLE
S D L
2309
W

FIG. 23

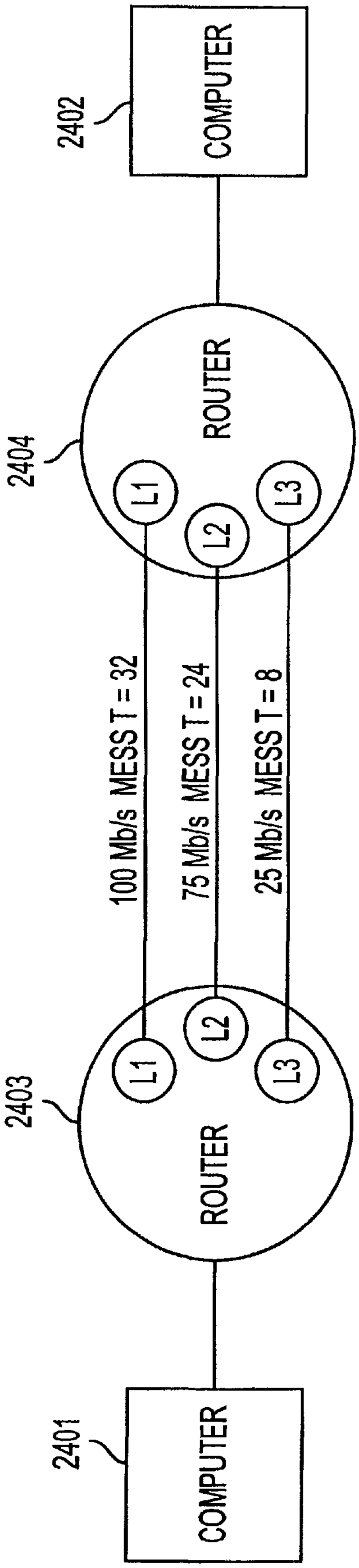

2402
COMPUTER
2404
ROUTER
L1
L2
L3
100 Mb/s MESS T = 32
75 Mb/s MESS T = 24
25 Mb/s MESS T = 8
L1
L2
L3
2403
ROUTER
2401
COMPUTER

FIG. 24

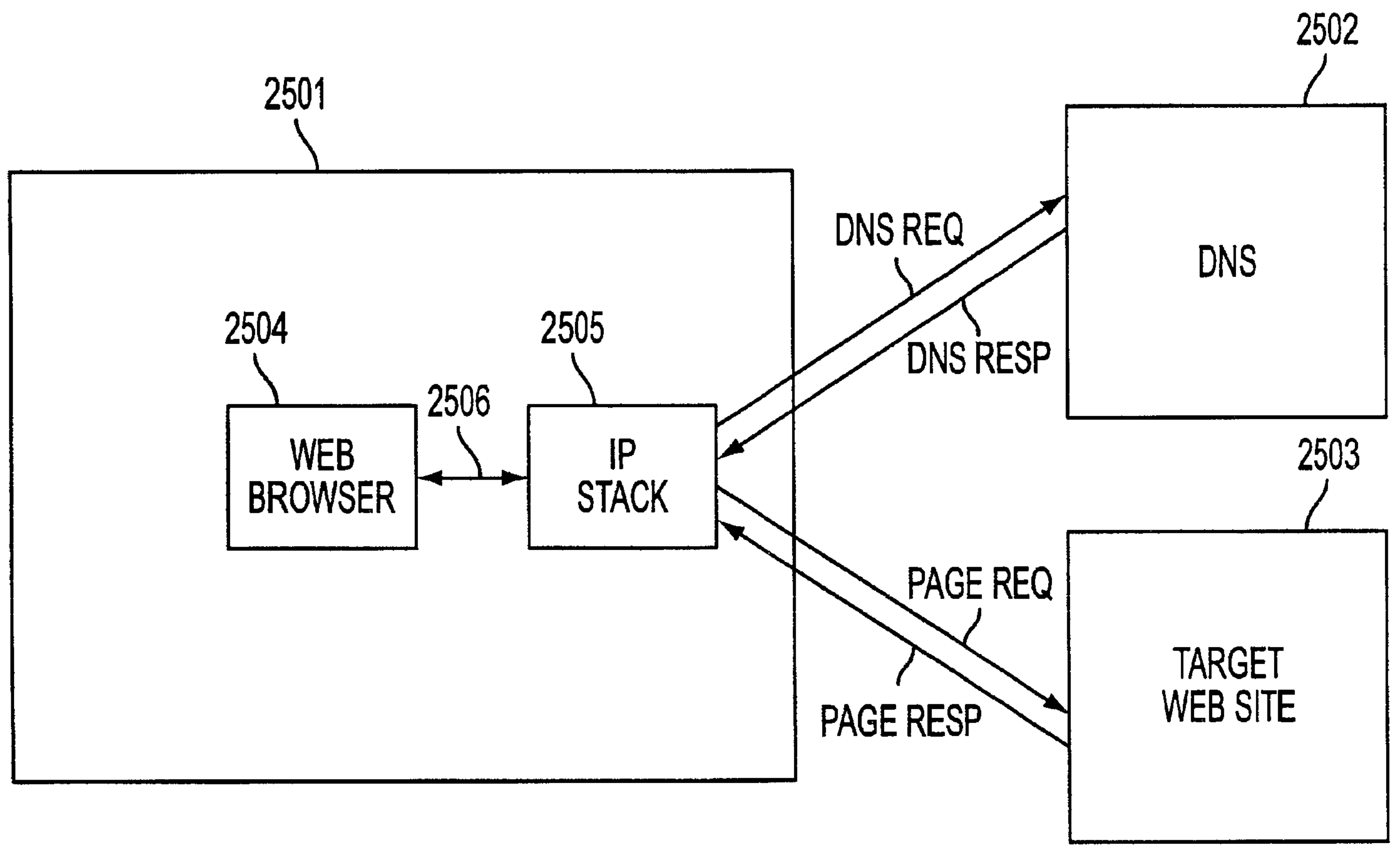


FIG. 25
(PRIOR ART)

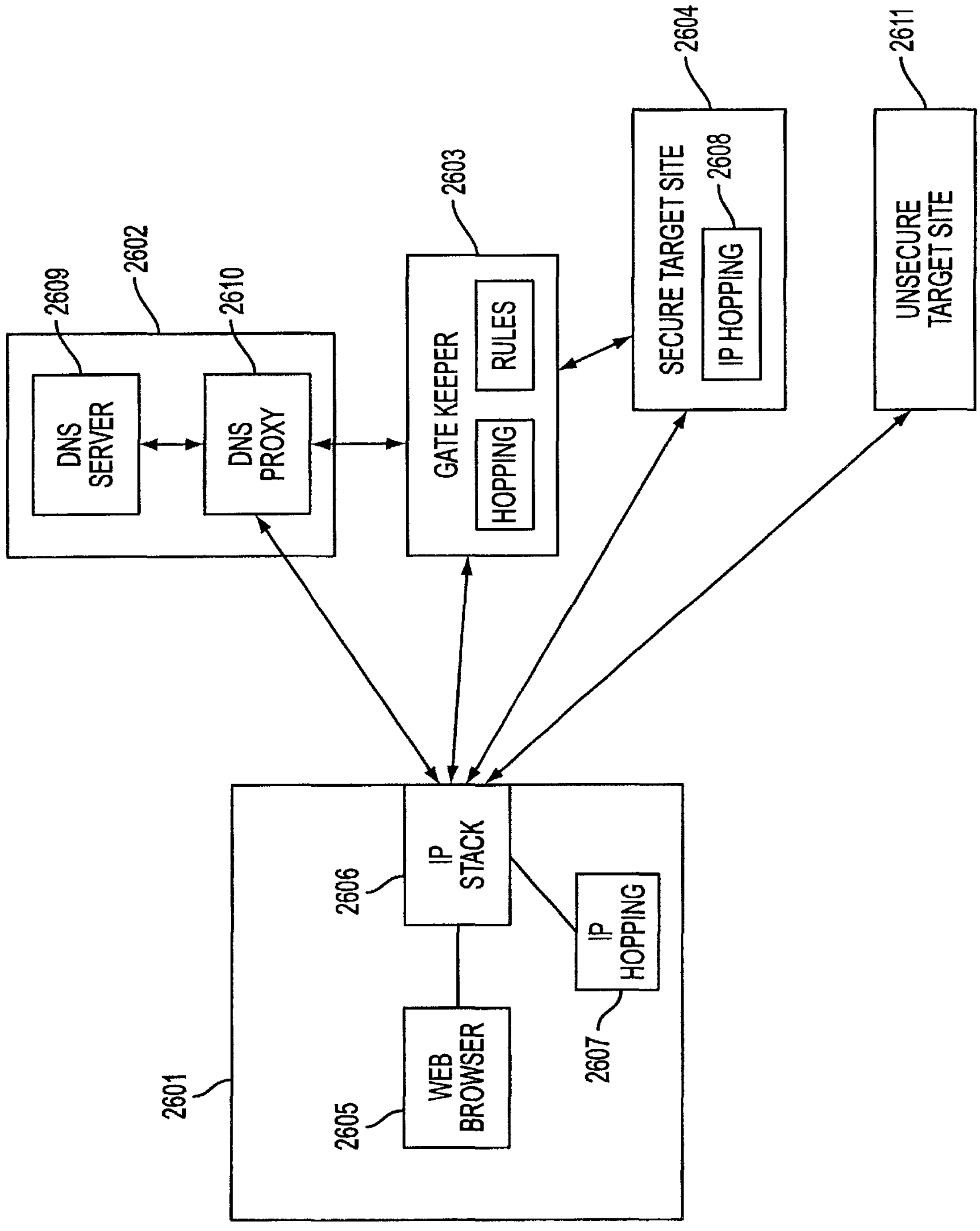

2602
2609
2610
DNS SERVER
DNS PROXY
2603
GATE KEEPER
HOPPING
RULES
2604
SECURE TARGET SITE
2608
IP HOPPING
2611
UNSECURE TARGET SITE
2601
2605
WEB BROWSER
2606
IP STACK
IP HOPPING
2607

FIG. 26

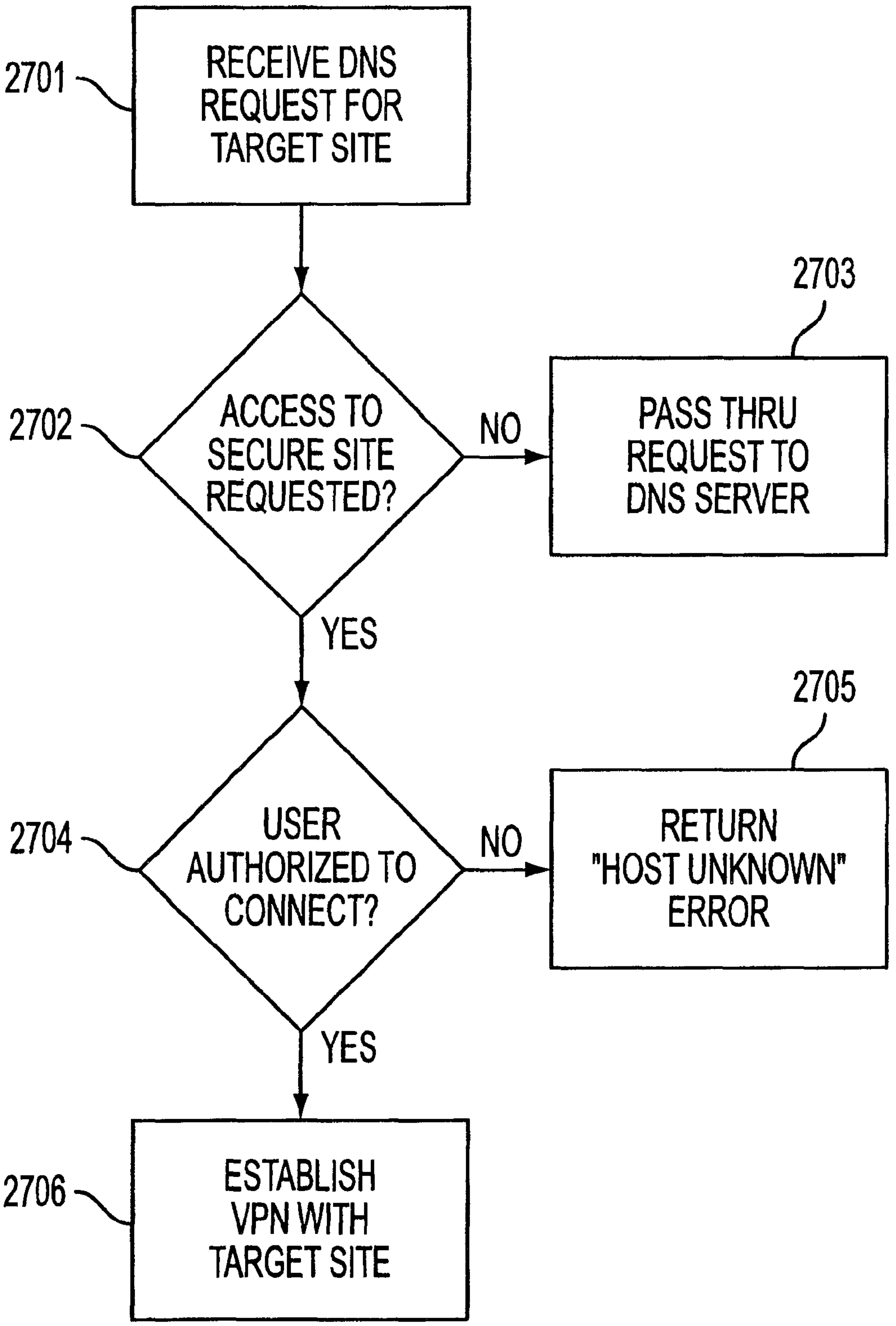

2701
RECEIVE DNS REQUEST FOR TARGET SITE
2702
ACCESS TO SECURE SITE REQUESTED?
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
2704
USER AUTHORIZED TO CONNECT?
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
2706
ESTABLISH VPN WITH TARGET SITE

FIG. 27

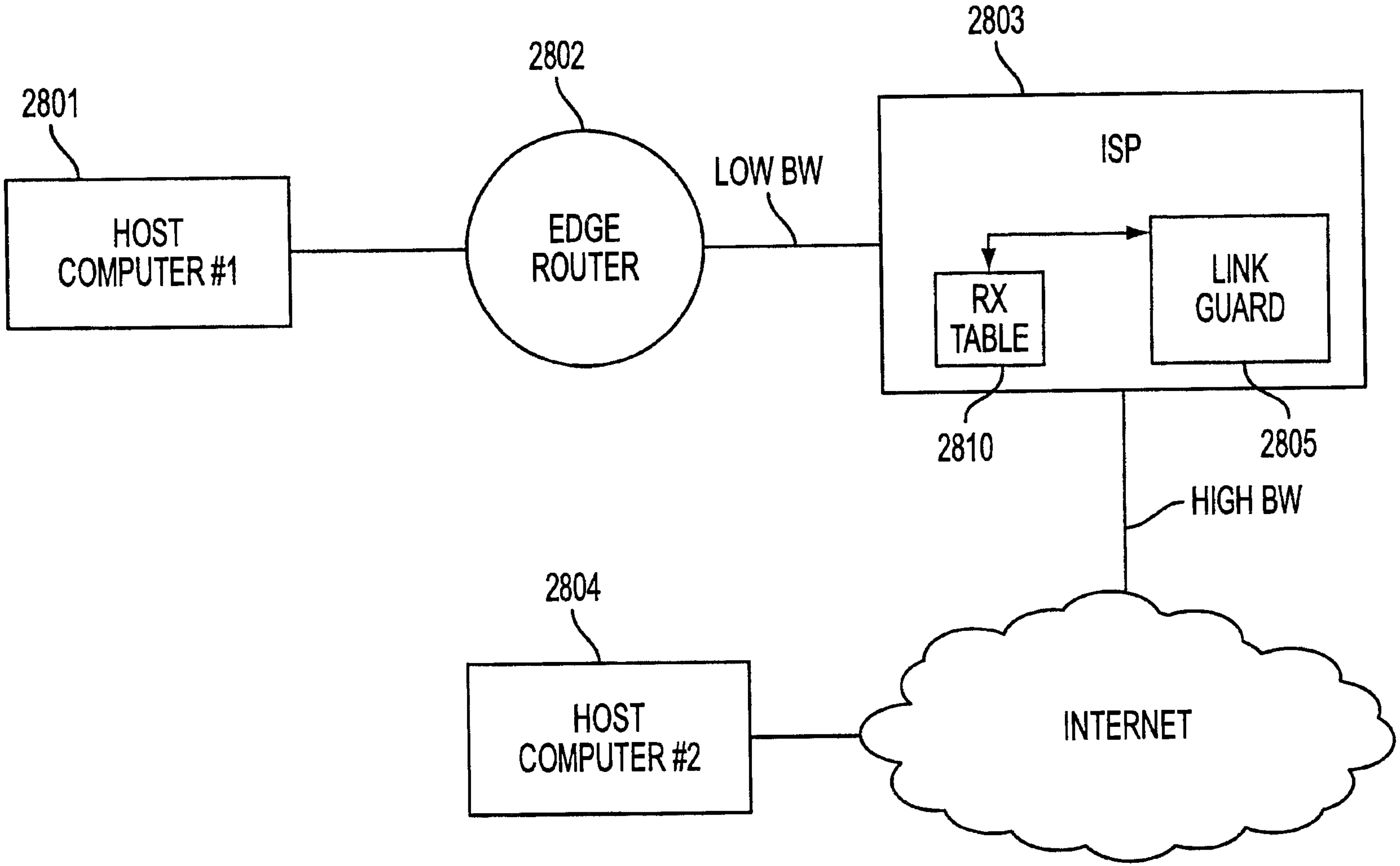

2801
HOST COMPUTER #1
2802
EDGE ROUTER
LOW BW
2803
ISP
RX TABLE
2810
LINK GUARD
2805
HIGH BW
2804
HOST COMPUTER #2
INTERNET

FIG. 28

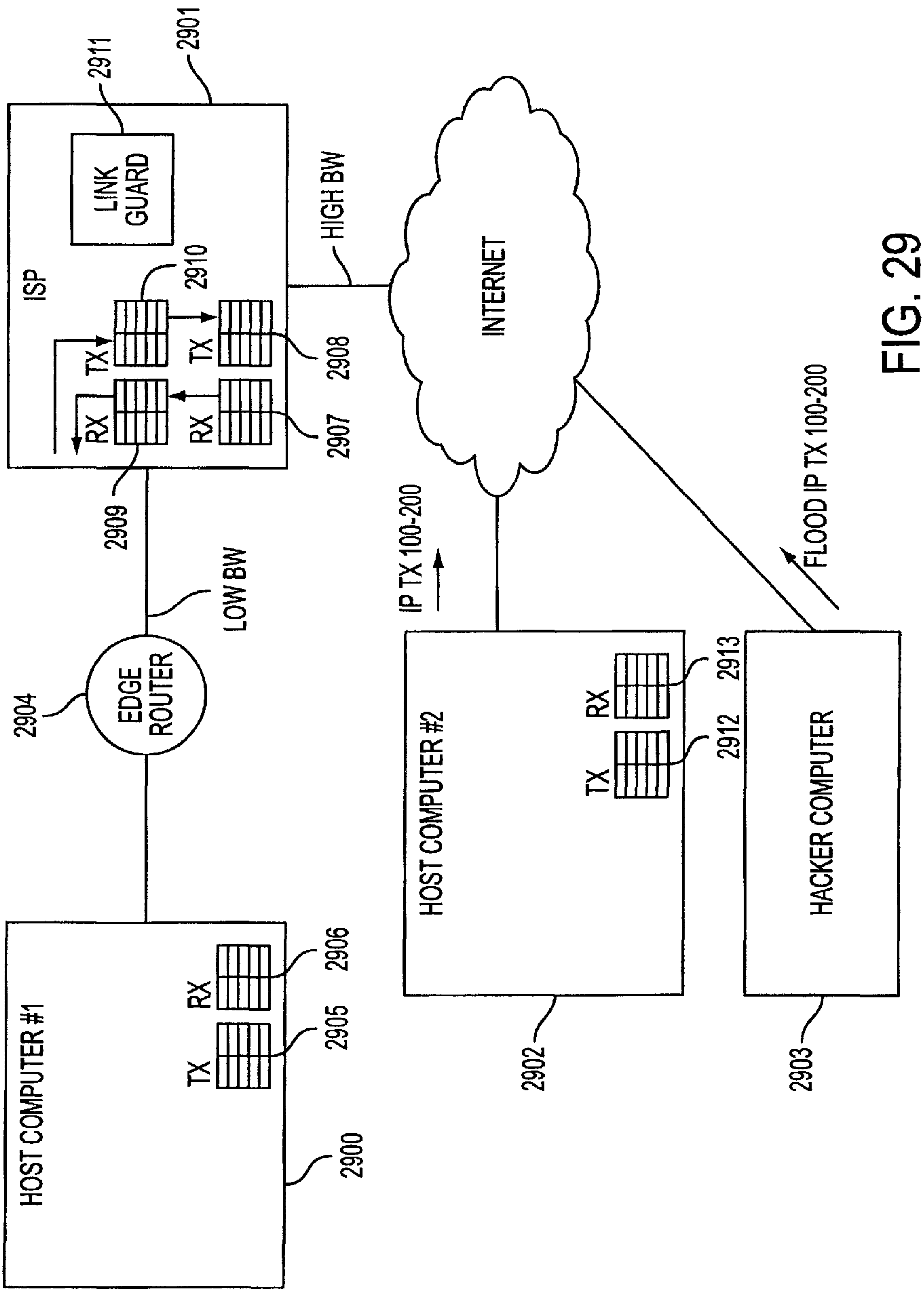

ISP
LINK GUARD
2911
2901
TX
TX
RX
RX
2910
2908
2907
2909
HIGH BW
LOW BW
INTERNET
EDGE ROUTER
2904
IP TX 100-200
FLOOD IP TX 100-200
HOST COMPUTER #1
TX
RX
2905
2906
2900
HOST COMPUTER #2
TX
RX
2912
2913
2902
HACKER COMPUTER
2903

FIG. 29

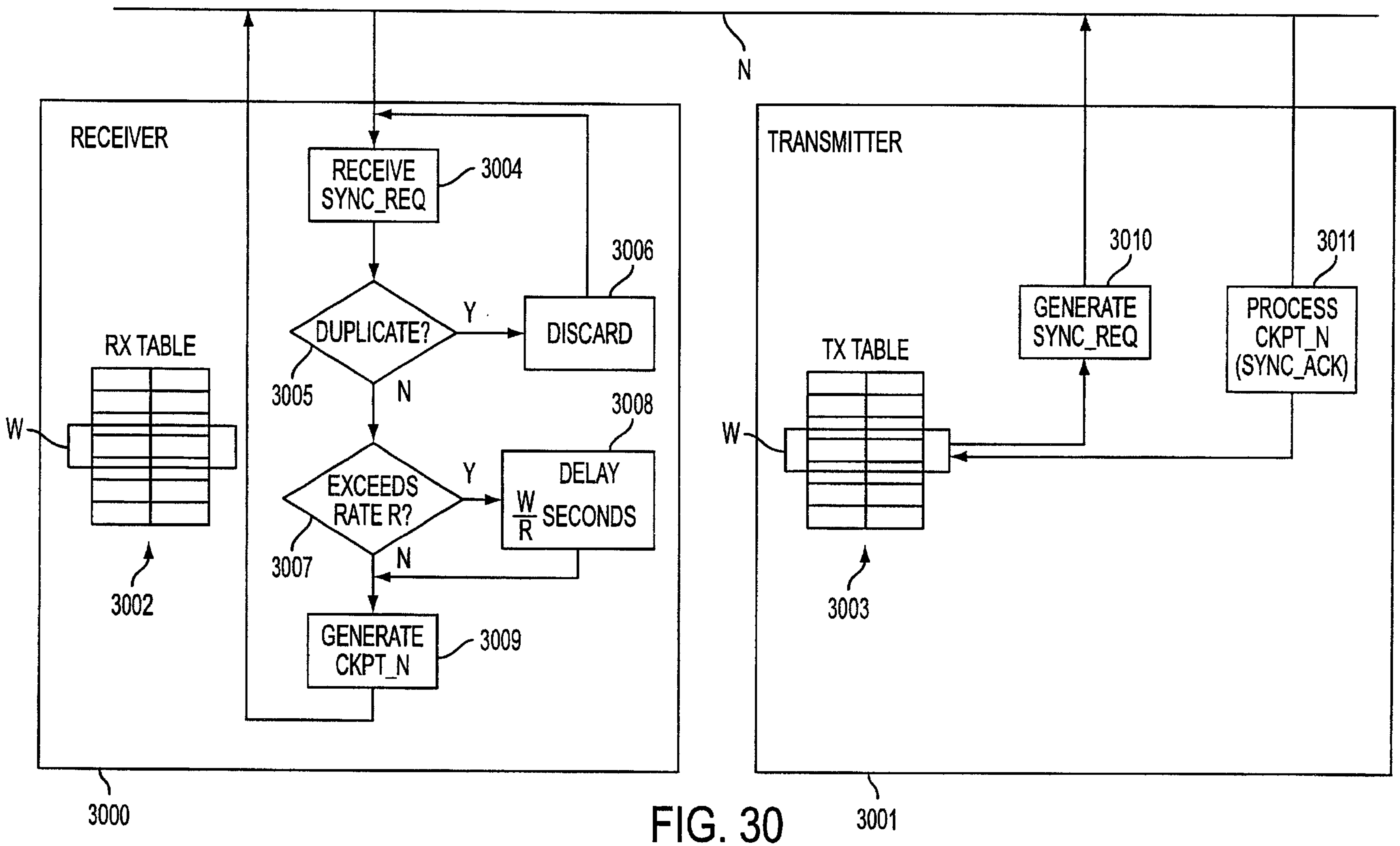

N
RECEIVER
RECEIVE SYNC_REQ
3004
3006
DUPLICATE?
Y
DISCARD
3005
N
3008
EXCEEDS RATE R?
Y
DELAY $\frac{W}{R}$ SECONDS
3007
N
GENERATE CKPT_N
3009
RX TABLE
W
3002
3000
TRANSMITTER
3010
GENERATE SYNC_REQ
3011
PROCESS CKPT_N (SYNC_ACK)
TX TABLE
W
3003
3001

FIG. 30

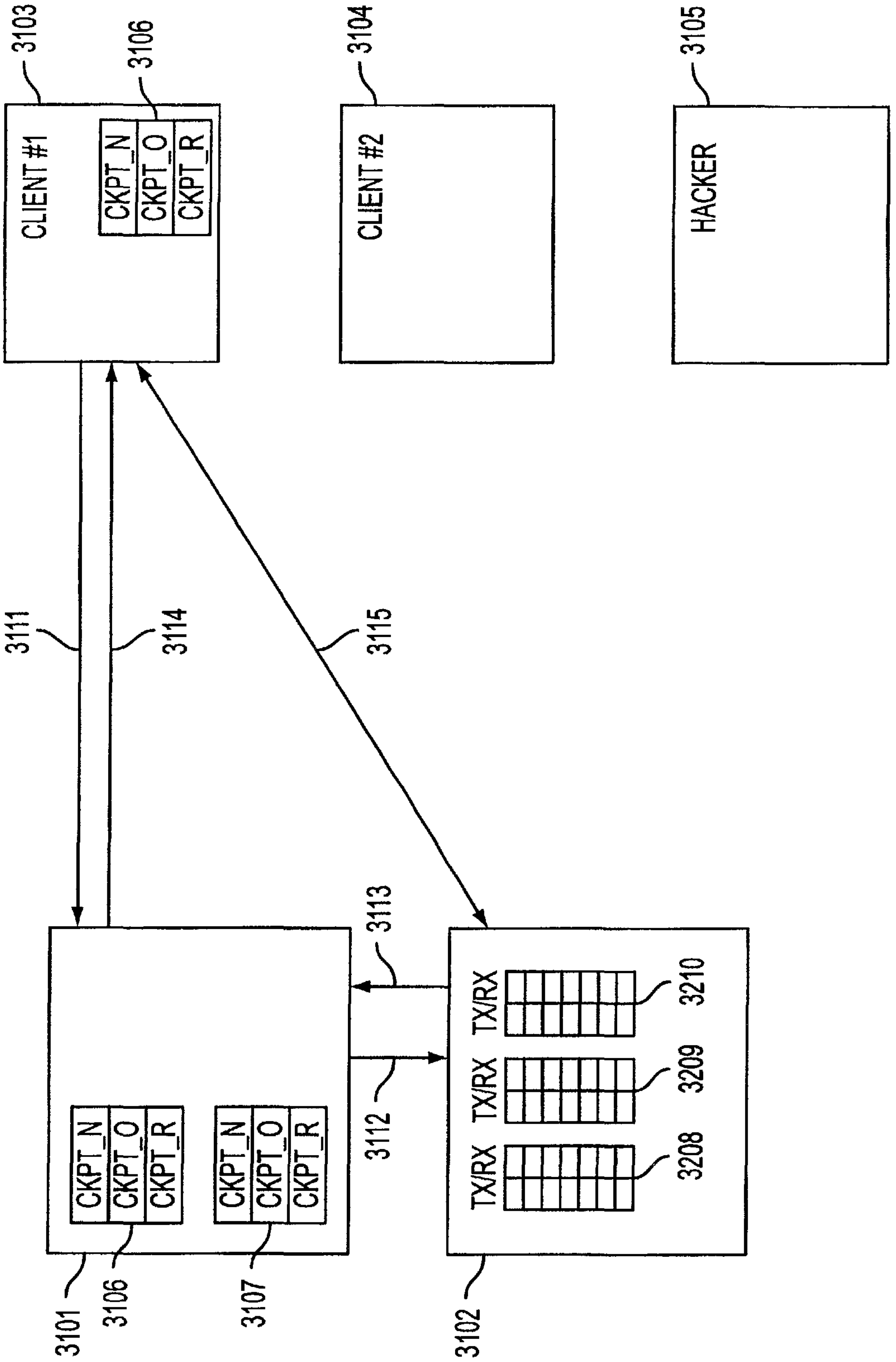

3103
3106
CLIENT #1
CKPT_N
CKPT_O
CKPT_R
3104
CLIENT #2
3105
HACKER
3111
3114
3115
3113
3112
CKPT_N
CKPT_O
CKPT_R
CKPT_N
CKPT_O
CKPT_R
TX/RX
TX/RX
TX/RX
3210
3209
3208
3101
3106
3107
3102

FIG. 31

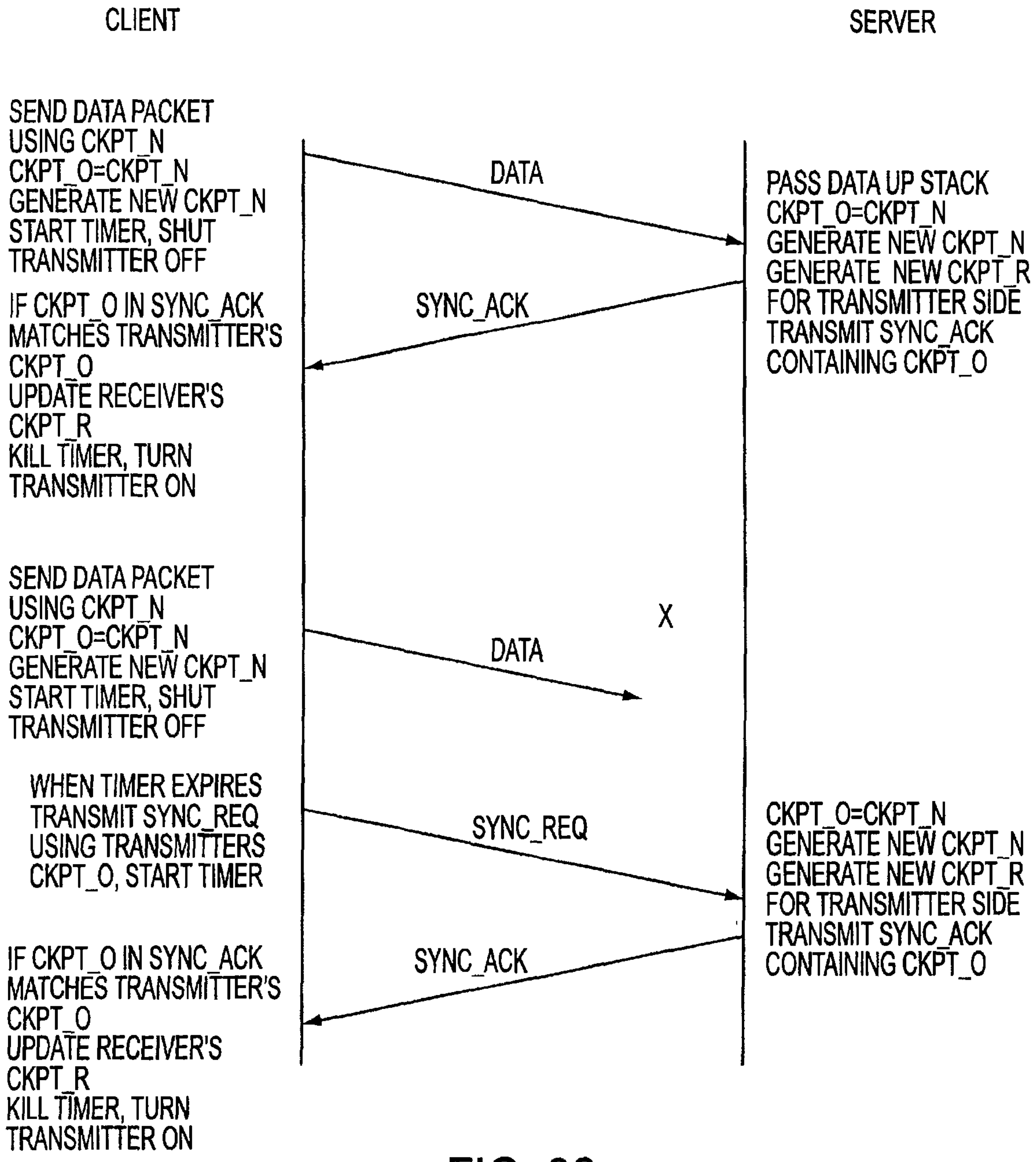

CLIENT
SERVER
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
DATA
PASS DATA UP STACK
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
X
DATA
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
CKPT_O, START TIMER
SYNC_REQ
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority from and is a continuation-in-part of previously filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999. The subject matter of that application, which is bodily incorporated herein, derives from provisional U.S. application No. 60/106,261 (filed Oct. 30, 1998) and No. 60/137,704 (filed Jun. 7, 1999).

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. **1**. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile

Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to

transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12A** shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12B** shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. **20** shows a system employing distributed transmission paths.

FIG. **21** shows a plurality of link transmission tables that can be used to route packets in the system of FIG. **20**.

FIG. **22**A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. **22**B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. **23** shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. **24** shows an example using the system of FIG. **23**.

FIG. **25** shows a conventional domain-name look-up service.

FIG. **26** shows a system employing a DNS proxy server with transparent VPN creation.

FIG. **27** shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. **28** shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. **29** shows one embodiment of a system employing the principles of FIG. **28**.

FIG. **30** shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. **31** shows a signaling server **3101** and a transport server **3102** used to establish a VPN with a client computer.

FIG. **32** shows message flows relating to synchronization protocols of FIG. **31**.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. **2**, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers **122–127** that are similar to regular IP routers **128–132** in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets **140**. TARP packets **140** are identical to normal IP packet messages that are routed by regular IP routers **128–132** because each TARP packet **140** contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's **140** IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal **110**. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet **140** since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal **110**.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key **146**. The link key **146** is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal **100** and the destination TARP terminal **110**. Each TARP router **122–127**, using the link key **146** it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key **146** and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router **122–127**, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet **140** to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet **140** to another TARP router **122–127** or to the destination TARP terminal **110**. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet **140** may forward the TARP packet **140** to the destination TARP terminal **110**. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet **140** may forward the TARP packet **140** to a TARP router **122–127** that the current TARP terminal chooses at random. As a result, each TARP packet **140** is routed through some minimum number of hops of TARP routers **122–127** which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP_C. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122–127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1**–**9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*–**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*–**207***c*, new TARP headers IP_T are formed. The TARP headers IP_T can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IP_T are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.

6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*–**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IP_T, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP_C is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal

110, or a TARP router **122–127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122–127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

1. SCALABILITY ENHANCEMENTS

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. **9**; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. **10**, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. FURTHER EXTENSIONS

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. **11**, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP1 and IP2, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP3. Each frame header generally includes a source hardware address **1101A** and a destination hardware address **1101B**; other well-known fields in frame headers are omitted from FIG. **11** for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. **12**A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. **12**A, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course-e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. **12A** and **12B** in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window **WI** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12A** (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101B** in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12B** shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node.

Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair; this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. 14):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead Capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers X_1, X_2, X_3 . . . X_k starting with seed X_0 using a recurrence

$$X_i=(aX_{i-1}+b)\bmod c, \qquad (1)$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1))\bmod c \qquad (2)$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1)\bmod c. \qquad (3)$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1)\bmod c=((a^i \bmod((a-1)c)(X_0(a-1)+b)-b)/(a-1))\bmod c \qquad (4).$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b) \bmod c$, b as b mod c and compute $a^i \bmod((a-1)c)$ (this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the j^{th} checkpoint, as X_0 and n as i, a node can store $a^n \bmod((a-1)c)$ once per LCR and set

$$X_{j+1}^w=X_{n(j+1)}=((a^n \bmod((a-1)c)(X_j^w(a-1)+b)-b)/(a-1))\bmod c, \qquad (5)$$

to generate the random number for the $j+1^{th}$ synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4)\text{mod } 15. \qquad (6)$$

If one sets X_0=1, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, c*(a−1)=15*30=450 and a^n mod((a−1)c)=31^3 mod(15*30)=29791 mod(450)=91. Equation (5) becomes:

$$((91(X_i30+4)-4)/30)\text{mod } 15 \qquad (7).$$

Table 1 shows the jump ahead calculations from (7) . The calculations start at 5 and jump ahead 3.

TABLE 1

			91	((91	
I	X_i	$(X_i30 + 4)$	$(X_i30 + 4) - 4$	$(X_i30 + 4) - 4)/30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^n-1). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number **135** using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector (s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. **16**). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2xWINDOW_SIZE+OoO active addresses (1≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧1). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. **17** depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2xWINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. **18** when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. **19** shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2xWINDOW_SIZE—OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. CONTINUATION-IN-PART IMPROVEMENTS

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22**A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path **X1** is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to asses the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times \text{MIN} + (1-\alpha) \times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1-\beta) \times P \quad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link **1** would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T

(32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.

5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

Asecond improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. 25. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. 25, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **2704**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **2703** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **2701**, a DNS look-up request is received for a target host. In step **2702**, a check is made to determine whether access to a secure host was requested. If not, then in step **2703** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **2702**, if access to a secure host was requested, then in step **2704** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **2705**). If the user has sufficient security privileges, then in step **2706** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc 2401], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid.

According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc 2104].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC_REQ every Ti seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of $M \times N \times W/R$ seconds after the last SYNC_REQ has been received and accepted, $2 \times M \times N \times W/R$ seconds after next to the last SYNC_REQ has been received and accepted, $C \times M \times N \times W/R$ seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer T1 noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.
2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and and other information (i.e user credentials) contained in the inner header. It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT R and transmits a SYNC ACK containing CKPT_O in its payload.
3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPT_O associated with the timer, it starts timer T1 noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.
5. When the server receives a SYNC_REQ on its CKPT_N it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new

CKPT_N. it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

What is claimed is:

1. A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:

(1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;

(2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and

(3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer.

2. The method of claim **1**, wherein steps (2) and (3) are performed at a DNS server separate from the client computer.

3. The method of claim **1**, further comprising the step of:

(4) in response to determining that the DNS request in step (2) is not requesting access to a secure target web site, resolving the IP address for the domain name and returning the IP address to the client computer.

4. The method of claim **1**, wherein step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to establish a VPN with the target computer and, if not so authorized, returning an error from the DNS request.

5. The method of claim **1**, wherein step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to resolve addresses of non secure target computers and, if not so authorized, returning an error from the DNS request.

6. The method of claim **1**, wherein step (3) comprises the step of establishing the VPN by creating an IP address hopping scheme between the client computer and the target computer.

7. The method of claim **1**, wherein step (3) comprises the step of using a gatekeeper computer that allocates VPN resources for communicating between the client computer and the target computer.

8. The method of claim **1**, wherein step (2) is performed in a DNS proxy server that passes through the request to a DNS server if it is determined in step (3) that access is not being requested to a secure target web site.

9. The method of claim **5**, wherein step (3) comprises the step of transmitting a message to the client computer to determine whether the client computer is authorized to establish the VPN target computer.

10. A system that transparently creates a virtual private network (VPN) between a client computer and a secure target computer, comprising:

a DNS proxy server that receives a request from the client computer to look up an IP address for a domain name, wherein the DNS proxy server returns the IP address for the requested domain name if it is determined that access to a non-secure web site has been requested, and wherein the DNS proxy server generates a request to create the VPN between the client computer and the secure target computer if it is determined that access to a secure web site has been requested; and

a gatekeeper computer that allocates resources for the VPN between the client computer and the secure web computer in response to the request by the DNS proxy server.

11. The system of claim **10**, wherein the gatekeeper computer creates the VPN by establishing an IP address hopping regime that is used to pseudorandomly change IP addresses in packets transmitted between the client computer and the secure target computer.

12. The system of claim **10**, wherein the gatekeeper computer determines whether the client computer has sufficient security privileges to create the VPN and, if the client computer lacks sufficient security privileges, rejecting the request to create the VPN.

13. A method of establishing communication between one of a plurality of client computers and a central computer that maintains a plurality of authentication tables each corresponding to one of the client computers, the method comprising the steps of:

(1) in the central computer, receiving from one of the plurality of client computers a request to establish a connection;

(2) authenticating, with reference to one of the plurality of authentication tables, that the request received in step (1) is from an authorized client;

(3) responsive to a determination that the request is from an authorized client, allocating resources to establish a virtual private link between the client and a second computer; and

(4) communicating between the authorized client and the second computer using the virtual private link.

14. The method of claim **13**, wherein step (4) comprises the step of communicating according to a scheme by which at least one field in a series of data packets is periodically changed according to a known sequence.

15. The method of claim **14**, wherein step (4) comprises the step of comparing an Internet Protocol (IP) address in a header of each data packet to a table of valid IP addresses maintained in a table in the second computer.

16. The method of claim **15**, wherein step (4) comprises the step of comparing the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window.

17. The method of claim **13**, wherein step (2) comprises the step of using a checkpoint data structure that maintains synchronization of a periodically changing parameter known by the central computer and the client computer to authenticate the client.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,502,135 B1
DATED : December 31, 2002
INVENTOR(S) : Edmund Colby Munger et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [56], **References Cited**, OTHER PUBLICATIONS, insert the following:

-- Search Report (dated 8/20/02), International Application No. PCT/US01/04340
Search Report (dated 8/23/02), International Application No. PCT/US01/13260
James E. Bellaire, "New Statement of Rules – Naming Internet Domains", Internet Newsgroup, July 30, 1995, 1 page.
D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, August 1, 1998, pages 22-25.
August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, Vol. 17, No. 4, 1998, pages 293-298.
Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of Information", Internet Newsgroup, June 21, 1997, 4 pages. --

Column 48,
Line 2, "VPN target computer" has been replaced with -- VPN with the target computer --.

Signed and Sealed this

Ninth Day of September, 2003

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006502135C1

(12) INTER PARTES REEXAMINATION CERTIFICATE (0271st)

United States Patent

Munger et al.

(10) **Number: US 6,502,135 C1**

(45) **Certificate Issued: Jun. 7, 2011**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY**

(75) Inventors: **Edmund Colby Munger**, Crownsville, MD (US); **Douglas Charles Schmidt**, Severna Park, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **Virnetx, Inc.**, Scotts Valley Drive, CA (US)

Reexamination Request:
No. 95/001,269, Dec. 8, 2009

Reexamination Certificate for:

Patent No.:	**6,502,135**
Issued:	**Dec. 31, 2002**
Appl. No.:	**09/504,783**
Filed:	**Feb. 15, 2000**

Certificate of Correction issued Sep. 9, 2003.

Related U.S. Application Data

(63) Continuation of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, and provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/225**; 709/229; 709/245

(58) **Field of Classification Search** 709/225
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,895,502	A	7/1959	Roper et al.
4,933,846	A	6/1990	Humphrey et al.
4,988,990	A	1/1991	Warrior
5,276,735	A	1/1994	Boebert et al.
5,303,302	A	4/1994	Burrows

(Continued)

FOREIGN PATENT DOCUMENTS

DE	199 24 575	12/1999
EP	0 814 589	12/1997
EP	836306 A1	4/1998
EP	0 838 930	4/1998
EP	0 858 189	8/1998

(Continued)

OTHER PUBLICATIONS

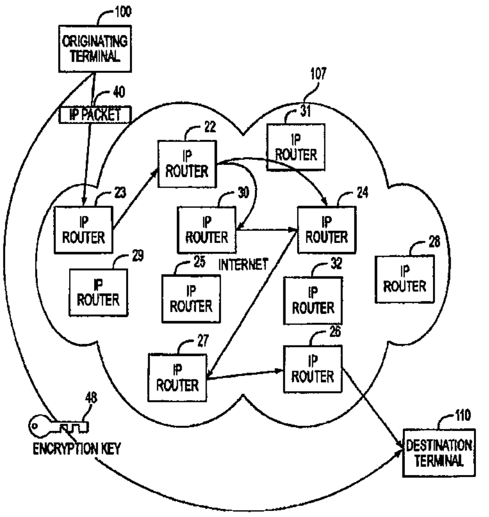
Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ssl3/draft302.txt on Feb. 4, 2002, 56 pages.

(Continued)

Primary Examiner—Andrew L Nalven

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.



U.S. PATENT DOCUMENTS

5,311,593	A	5/1994	Carmi
5,329,521	A	7/1994	Walsh et al.
5,341,426	A	8/1994	Barney et al.
5,367,643	A	11/1994	Chang et al.
5,384,848	A	1/1995	Kikuchi
5,511,122	A	4/1996	Atkinson
5,559,883	A	9/1996	Williams
5,561,669	A	10/1996	Lenney et al.
5,588,060	A	12/1996	Aziz
5,625,626	A	4/1997	Umekita
5,629,984	A	5/1997	McManis
5,654,695	A	8/1997	Olnowich et al.
5,682,480	A	10/1997	Nakagawa
5,689,566	A	11/1997	Nguyen
5,740,375	A	4/1998	Dunne et al.
5,764,906	A	6/1998	Edelstein et al.
5,771,239	A	6/1998	Moroney et al.
5,774,660	A	6/1998	Brendel et al.
5,787,172	A	7/1998	Arnold
5,796,942	A	8/1998	Esbensen
5,805,801	A	9/1998	Holloway et al.
5,805,803	A	9/1998	Birrell et al.
5,822,434	A	10/1998	Caronni et al.
5,842,040	A	11/1998	Hughes et al.
5,845,091	A	12/1998	Dunne et al.
5,864,666	A	1/1999	Shrader
5,867,650	A	2/1999	Osterman
5,870,610	A	2/1999	Beyda et al.
5,878,231	A	3/1999	Baehr et al.
5,892,903	A	4/1999	Klaus
5,898,830	A	4/1999	Wesinger et al.
5,905,859	A	5/1999	Holloway et al.
5,918,019	A	6/1999	Valencia
5,950,195	A	9/1999	Stockwell et al.
5,996,016	A	11/1999	Thalheimer et al.
6,006,259	A	12/1999	Adelman et al.
6,006,272	A	12/1999	Aravamudan et al.
6,016,318	A	1/2000	Tomoike
6,016,512	A	1/2000	Huitema
6,041,342	A	3/2000	Yamaguchi
6,052,788	A	4/2000	Wesinger et al.
6,055,574	A	4/2000	Smorodinsky et al.
6,061,346	A	5/2000	Nordman
6,061,736	A	5/2000	Rochberger et al.
6,079,020	A	6/2000	Liu
6,081,900	A	6/2000	Subramaniam et al.
6,092,200	A	7/2000	Muniyappa et al.
6,101,182	A	8/2000	Sistanizadeh et al.
6,119,171	A	9/2000	Alkhatib
6,119,234	A	9/2000	Aziz et al.
6,147,976	A	11/2000	Shand et al.
6,157,957	A	12/2000	Berthaud
6,158,011	A	12/2000	Chen et al.
6,168,409	B1	1/2001	Fare
6,173,399	B1	1/2001	Gilbrech
6,175,867	B1	1/2001	Taghadoss
6,178,409	B1	1/2001	Weber et al.
6,178,505	B1	1/2001	Schneider et al.
6,179,102	B1	1/2001	Weber et al.
6,199,112	B1	3/2001	Wilson
6,202,081	B1	3/2001	Naudus
6,222,842	B1	4/2001	Sasyan et al.
6,223,287	B1	4/2001	Douglas et al.
6,226,748	B1	5/2001	Bots et al.
6,226,751	B1	5/2001	Arrow et al.
6,233,618	B1	5/2001	Shannon
6,243,360	B1	6/2001	Basilico
6,243,749	B1	6/2001	Sitaraman et al.
6,243,754	B1	6/2001	Guerin et al.
6,246,670	B1	6/2001	Karlsson et al.
6,256,671	B1	7/2001	Strentzsch et al.
6,262,987	B1	7/2001	Mogul
6,263,445	B1	7/2001	Blumenau
6,286,047	B1	9/2001	Ramanathan et al.
6,298,341	B1	10/2001	Mann et al.
6,301,223	B1	10/2001	Hrastar et al.
6,308,274	B1	10/2001	Swift
6,311,207	B1	10/2001	Mighdoll et al.
6,314,463	B1	11/2001	Abbott et al.
6,324,161	B1	11/2001	Kirch
6,330,562	B1	12/2001	Boden et al.
6,332,158	B1	12/2001	Risley et al.
6,333,272	B1	12/2001	McMillin et al.
6,338,082	B1	1/2002	Schneider
6,353,614	B1	3/2002	Borella et al.
6,430,155	B1	8/2002	Davie et al.
6,430,610	B1	8/2002	Carter
6,487,598	B1	11/2002	Valencia
6,502,135	B1	12/2002	Munger et al.
6,505,232	B1	1/2003	Mighdoll et al.
6,510,154	B1	1/2003	Mayes et al.
6,549,516	B1	4/2003	Albert et al.
6,557,037	B1	4/2003	Provino
6,571,296	B1	5/2003	Dillon
6,571,338	B1	5/2003	Shaio et al.
6,581,166	B1	6/2003	Hirst et al.
6,618,761	B2	9/2003	Munger et al.
6,671,702	B2	12/2003	Kruglikov et al.
6,687,551	B2	2/2004	Steindl
6,687,746	B1	2/2004	Shuster et al.
6,701,437	B1	3/2004	Hoke et al.
6,714,970	B1	3/2004	Fiveash et al.
6,717,949	B1	4/2004	Boden et al.
6,752,166	B2	6/2004	Lull et al.
6,757,740	B1	6/2004	Parekh et al.
6,760,766	B1	7/2004	Sahlqvist
6,826,616	B2	11/2004	Larson et al.
6,839,759	B2	1/2005	Larson et al.
6,937,597	B1	8/2005	Rosenberg et al.
7,010,604	B1	3/2006	Munger et al.
7,039,713	B1	5/2006	Van Gunter et al.
7,072,964	B1	7/2006	Whittle et al.
7,133,930	B2	11/2006	Munger et al.
7,167,904	B1	1/2007	Devarajan et al.
7,188,175	B1	3/2007	McKeeth
7,188,180	B2	3/2007	Larson et al.
7,197,563	B2	3/2007	Sheymov et al.
7,353,841	B2	4/2008	Kono et al.
7,461,334	B1	12/2008	Lu et al.
7,490,151	B2	2/2009	Munger et al.
7,493,403	B2	2/2009	Shull et al.
2001/0049741	A1	12/2001	Skene et al.
2002/0004898	A1	1/2002	Droge
2004/0199493	A1	10/2004	Ruiz et al.
2004/0199520	A1	10/2004	Ruiz et al.
2004/0199608	A1	10/2004	Rechterman et al.
2004/0199620	A1	10/2004	Ruiz et al.
2005/0055306	A1	3/2005	Miller et al.
2007/0208869	A1	9/2007	Adelman et al.
2007/0214284	A1	9/2007	King et al.
2007/0266141	A1	11/2007	Norton
2008/0235507	A1	9/2008	Ishikawa et al.

FOREIGN PATENT DOCUMENTS

GB	2 317 792	4/1998
GB	2 334 181 A	8/1999
JP	62-214744	9/1987
JP	04-363941	12/1992
JP	09-018492	1/1997
JP	10-070531	3/1998
WO	WO 9827783 A	6/1998

WO	WO 98/27783	6/1998
WO	WO 98 55930	12/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/17775	3/2000
WO	WO 001/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01/016766	3/2001
WO	WO 01 50688	7/2001

OTHER PUBLICATIONS

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293–298.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278–375.

D. Clark, "US Calls for Private Domain–Name System", Computer, IEEE Computer Society, Aug. 1, 1998, pp. 22–25.

Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Work–shop, ISW'99. Proceedings (Lecture Springer–Verlag Berlin, Germany, [Online] 1999, pp. 85–102, XP002399276, ISBN 3–540–666.

Dolev, Shlomi and Ostrovsky, Rafil, "Efficient Anonymous Multicast and Reception" (Extended Abstact), 16 pages.

Donald E. Eastlake, 3rd, "Domain Name System Security Extensions", Internet Draft, Apr. 1998, pp. 1–51.

F. Halsall, "Data Communications, Computer Networks and Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198–203.

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security" Protection of Location Information in Mobile IP, IEEE publication, 1996, pp. 963–967.

Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan__trees/freeswan–1.3/doc/glossary.html on Feb. 21, 2002, 25 pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan__trees/freeswan–1.3/doc/rationale.html on Feb. 21, 2002, 4 pages.

James E. Bellaire, "New Statement of Rules–Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation. 2000, pp. 1–14.

Laurie Wells (LANCASTERBIBELMAIL MSN COM); "Subject: Security Icon" USENET Newsgroup, Oct. 19, 1998, XP002200606, 1 page.

Linux FreeS/WAN Index File, printed from http://liberty.freewan.org/freeswan__trees/freeswan–1.3/doc/ on Feb. 21, 2002, 3 Pages.

P. Srisuresh et al., "DNS extensions to Network address Translators (DNS_ALG)", Internet Draft, Jul. 1998, pp. 1–27.

RFC 2401 (dated Nov. 1998) Security Architecture for the Internet Protocol (RTP).

RFC 2543–SIP (dated Mar. 1999): Session Initiation Protocol (SIP or SIPS).

Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82–94.

Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

Search Report (dated Oct. 7, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dataed Nov. 13, 2002), International Application No. PCT/US01/04340.

Search Report, IPER (dated Feb.6, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.

Sankar, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conferece on Communications architectures & protocols. pp. 84–91, ACM Press, NY, NY 1986.

Shree Murthy et al., "Congestion–Oriented Shortest Multi–path Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028–1036.

W. Stallings, "Cryptography And Network Security", 2nd, Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399–440.

Fasbender, A. et al., Variable and Scalable Security: Protection of Location Information in Mobile IP, IEEE VTS, 46th, 1996, 5 pp.

156. Finding Your Way Through the VPN Maze (1999) ("PGP").

WatchGuard Technologies, Inc., WatchGuard LiveSecurity for MSS Powerpoint (Feb. 14 2000) (resubmitted).

WatchGuard Technologies, Inc., MSS Version 2.5, Add–On for WatchGuard SOHO Release Notes (Jul. 21, 2000).

Yuan Dong Feng, "A novel scheme combining interleaving technique with cipher in Rayleigh fading channels," Proceedings of the International Conference on Communication technology, 2:S47–02–1–S47–02–4 (1998).

D.W. Davies and W.L. Price, edited by Tadahiro Uezona, "Network Security", Japan, Nikkei McGraw–Hill, Dec. 5, 1958, First Edition, first copy, p. 102–108.

U.S. Appl. No. 60/134,547 filed May 17, 1999, Victor Sheymov.

U.S. Appl. No. 60/151,563 filed Aug. 31, 1999, Bryan Whittles.

U.S. Appl. No. 09/399,753 filed Sep. 22, 1998, Graig Miller et al.

Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009, *VirnetX Inc. and Science Applications International Corp.* v. *Microsoft Corporation.*

Appendix A of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

Concordance Table For the References Cited in Tables on pp. 6–15, 71–80 and 116–124 of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

I. P. Mockapetris, "DNS Encoding of Network Names and Other Types," Network Working Group, RFC 1101 (Apr. 1989) (RFC1101, DNS SRV).

DNS–related corresponding dated Sep. 7, 1993 to Sep. 20, 1993. (Pre KX, KX Records).

R. Atkinson, "An Internetwork Authentication Architecture," Naval Research Laboratory, Center for High Assurance Computing Systems (Aug. 5, 1993). (Atkinson NRL, KX Records).

Henning Schulzrinne, *Personal Mobility For Multimedia Services In The Internet,* Proceedings of the Interactive Distributed Multimedia Systems and Services European Workshop at 143 (1996) (Schulzrinne 96).

Microsoft Corp., *Microsoft Virtual Private Networking: Using Point-to-Point Tunneling Protocol for Low-Cost, Secure, Remote Access Across the Internet* (1996) (printed from 1998 PDC DVD-ROM) (Point to Point, Microsoft Prior Art VPN Technology).

"Safe Surfing: How to Build a Secure World Wide Web Connection," IBM Technical Support Organization, (Mar. 1996). (Safe Surfing, Website Art).

Goldschlag, et al., "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK (May 1996). (Goldschlag II, Onion Routing).

"IPSec Minutes From Montreal", IPSEC Working Group Meeting Notes, http://www.sandleman.ca/ipsec/1996/08/msg00018.html (Jun. 1996). (IPSec Minutes, FreeS/WAN).

J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose, California, Jul. 1996. (Galvin, DNSSEC).

J. Gilmore, et al. "Re: Key Management, anyone? (DNS Keying)," IPSec Working Group Mailing List Archives (Aug. 1996). (Gilmore DNS, FreeS/WAN).

H. Orman, et al."Re: Re: DNS? was Re: Key Management, anyone?" IETF IPSec Working Group Mailing List Archive (Aug. 1996/Sep. 1996). (Orman DNS, FreeS/WAN).

Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services* (*DNS SRV*), IETF RFC 2052 (Oct. 1996). (RFC 2052, DNS SRV).

Freier, et al. "The SSL Protocol Version 3.0," Transport Layer Security Working Group (Nov. 18, 1996). (SSL, Underlying Security Technology).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 2, 1996). (RFC 2543 Internet Draft 1).

M.G. Reed, et al. "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference, San Diego, CA, Dec. 9-13, 1996. (Reed, Onion Routing).

Kenneth F. Alden & Edward P. Wobber, *The AltaVista Tunnel: Using the Internet to Extend Corporate Networks,* Digital Technical Journal (1997) (Alden, AltaVista.

Automative Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).

Automative Industry Action Group, "ANX Release 1 Draft Document Publication," AIAG Publications (1997). (AIAG Release, ANX).

Aventail Corp., "AutoSOCKS v. 2.1 Datasheet," available at http://www.archive.org/web/19970212013409/www.aventail.com/prod/autosk2ds.html (1997). (AutoSOCKS, Aventail).

Aventail Corp. "Aventail VPN Data Sheet," available at http://www.archive.org/web/19970212013043/www.aventail.com/prod/vpndata.html (1997). (Data Sheet, Aventail).

Aventail Corp., "Directed VPN Vs. Tunnel," available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/directvpn.html (1997). (Directed VPN, Aventail).

Aventail Corp., "Managing Corporate Access to the Internet," Aventail AutoSOCKS White Paper available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/ipmwp.html (1997). (Corporate Access, Aventail).

Aventail Corp., "Socks Version 5," Aventail Whitepaper, available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/sockswp.html (1997). (Socks, Aventail).

Aventail Corp., "VPN Server V2.0 Administration Guide," (1997). (VPN, Aventail).

Goldschlag, et al. *"Privacy on the Internet,"* Naval Research Laboratory, Center for High Assurance Computer Systems (1997). (Goldschlag I, Onion Routing).

Microsoft Corp., *Installing Configuring and Using PPTP with Microsoft Clients and Servers* (1997). (Using PPTP, Microsoft Prior Art VPN Technology).

Microsoft Corp., *IP Security for Microsoft Windows NT Server 5.0* (1997) (printed from 1998 PDC DVD-ROM). (IP Security, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Microsoft Windows NT Active Directory: An Introduction to the Next Generation Directory Services* (1997) (printed from 1998 PDC DVD-ROM). (Directory, Microsoft Prior Art VPN Technology).

Microsoft Corp. *Routing and Remote Access Service for Windows NT Server NewOpportunities Today and Looking Ahead* (1997) (printed from 1998 PDC DVD-ROM). (Routing, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Understanding Point-to-Point Tunneling Protocol PPTP* (1997) (printed from 1998 PDC DVD-ROM). (Understanding PPTP, Microsoft Prior Art VPN Technology).

J. Mark Smith et al., *Protecting a Private Network: The AltaVista Firewall,* Digital Technical Journal (1997). (Smith, AltaVista).

Naganand Doraswamy *Implementation of Virtual Private Networks* (*VPNs*) *with IPSecurity,* <draft-ietf-ipsec-vpn-00.txt> (Mar. 12, 1997). (Doraswamy).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Mar. 27, 1997). (RFC 2543 Internet Draft 2).

Aventail Corp., "Aventail and Cybersafe to Provide Secure Authentication For Internet and Intranet Communication," Press Release, Apr. 3, 1997. (Secure Authentication, Aventail).

D. Wagner, et al. "Analysis of the SSL 3.0 Protocol," (Apr. 15, 1997). (Analysis, Underlying Security Technologies).

Automotive Industry Action Group, "ANXO Certification Authority Service and Directory Service Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Definition, ANX).

Automotive Industry Action Group, "ANXO Certification Process and ANX Registration Process Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Certification, ANX).

Aventail Corp. "Aventail Announces the First VPN Solution to Assure Interoperability Across Emerging Security Protocols," Jun. 2, 1997. (First VPN, Aventail).

Syverson, et al. "Private Web Browsing," Naval Research Laboratory, Center for High 8 Assurance Computer Systems (Jun. 2, 1997). (Syverson, Onion Routing).

Bellcore, "Metrics, Criteria, and Measurement Technique Requirements for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (Jun. 16, 1997). (AIAG Requirements, ANX).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 31, 1997). (RFC 2543 Internet Draft 3).

R. Atkinson, "Key Exchange Delegation Record for the DNS," Network Working Group, RFC 2230 (Nov. 1997). (RFC 2230, KX Records).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 11, 1997). (RFC 2543 Internet Draft 4).

1998 Microsoft Professional Developers Conference DVD ("1998 PDC DVD–ROM") (including screenshots captured therefrom and produced as MSFTVX 00018827–00018832). (Conference, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Virtual Private Networking An Overview* (1998) (printed from 1998 PDC DVD–ROM) (Overview, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Windows NT 5.0 Beta Has Public Premiere at Seattle Mini–Camp Seminar attendees get first look at the performance and capabilities of Windows NT 5.0* (1998) (available at hap //www.microsoft.com/presspass/features/1998/10–19nt5.mspxpftrue). (NT Beta, Microsoft Prior Art VPN Technology).

"What ports does SSL use" available at stason.org/TULARC/security/ssl–talk/3–4–What–ports–does–ssl–use.html (1998). (Ports, DNS SRV).

Aventail Corp., "Aventail VPN V2.6 Includes Support for More Than Ten Authentication Methods Making Extranet VPN Development Secure and Simple," Press Release, Jan. 19, 1998. (VPN V2.6, Aventail).

R. G. Moskowitz, "Network Address Translation Issues with IPsec," Internet Draft, Internet Engineering Task Force, Feb. 6, 1998. (Moskowitz).

H. Schulzrinne, et al, "Internet Telephony Gateway Location," Proceedings of IEEE INfocom '98, The Conference on Computer Communications, vol. 2 (Mar. 29–Apr. 2, 1998). (Gateway, Schulzrinne).

C. Huitema, 45 al. "Simple Gateway Control Protocol," Version 1.0 (May 5, 1998). (SGCP).

DISA "Secret Internet Protocol Router Network," SIPRNET Program Management Office (D3113) DISN Networks, DISN Transmission Services (May 8, 1998). (DISA, SIPRNET).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (May 14, 1998). (RFC 2543 Internet Draft 5).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jun. 17, 1998). (RFC 2543 Internet Draft 6).

D. McDonald, et al. "PF_KEY Management API, Version 2," Network Working Group, RFC 2367 (Jul. 1998). (RFC 2367).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 16, 1998). (RFC 2543 Internet Draft 7).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Aug. 7, 1998). (RFC 2543 Internet Draft 8).

Microsoft Corp., *Company Focuses on Quality and Customer Feedback* (Aug. 18, 1998). (Focus, Microsoft Prior Art VPN Technology).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Sep 18, 1998). (RFC 2543 Internet Draft 9).

Atkinson, et al. "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998). (RFC 2401, Underlying Security Technologies).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 12, 1998). (RFC 2543 Internet Draft 10) 9.

Donald Eastlake, *Domain Name System Security Extensions,* IETF–DNS Security Working Group (Dec. 1998). (DNSSEC–7).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 15, 1998). (RFC 2543 Internet Draft 11).

Aventail Corp., "Aventail Connect 3.1/2.6Administrator's Guide," (1999). (Aventail Administrator 3.1, Aventail).

Aventail Corp., "Aventail Connect 3.1/2.6 User's Guide," (1999). (Aventail Administrator 3.1, Aventail).

Aventail Corp., "Aventail ExtraWeb Server v3.2 Administrator's Guide," (1999). (Aventail ExtraWeb 3.2, Aventail).

Kaufman et al, "Implementing IPsec," (Copyright 1999). (Implementing IPSEC, VPN References).

Network Solutions, Inc. "Enabling SSL," NSI Registry (1999). (Enabling SSL, Underlying Security Technologies).

Check Point Software Technologies Ltd. (1999) (Check Point, Checkpoint FW).

Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services* (*DNS SRV*), <draft–ietf–dnsind–frc2052bis–02.txt> (Jan. 1999). (Gulbrandsen 99, DNS SRV).

C. Scott, et al. *Virtual Private Networks,* O'Reilly and Associates, Inc., 2nd ed. (Jan. 1999). (Scott VPNs).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jan. 15, 1999). (RFC 2543 Internet Draft 12).

Goldschlag, et al., "Onion Routing for Anonymous and Private Internet Connections," Naval Research Laboratory, Center for High Assurance Computer Systems (Jan. 28, 1999). (Goldschlag III, Onion Routing).

H. Schulzrinne, "Internet Telephony: architecture and protocols—an IETF perspective," Computer Networks, vol. 31, No. 3 (Feb. 1999). (Telephony, Schulzrinne).

M. Handley, et al, "SIP: Session Initiation Protocol," Network Working Group, RFC 2543 and Internet Drafts (Dec. 1996–Mar. 1999). (Handley, RFC 2543).

FreeS/WAN Project, *Linux FreeS/WAN Compatibility Guide* (Mar. 4, 1999). (FreeS/WAN Compatibility Guide, FreeS/WAN).

Telcordia Technologies, "ANX Release 1 Document Corrections," AIAG (May 11, 1999). (Telcordia, ANX).

Ken Hornstein & Jeffrey Altman, *Distributing Kerberos KDC and Realm Information with DNS* <draft–eitf–cat–krb–dns–locate–oo.txt> (Jun. 21, 1999). (Hornstein, DNS SRV).

Bhattacharya et al. "An LDAP Schema for Configuration and Administration of IPSec Based Virtual Private Networks (VPNs)", IETF Internet Draft (Oct. 1999). (Bhattcharya LDAP VPN).

B. Patel, et al, "DHCP Configuration of IPSEC Tunnel Mode," IPSEC Working Group, Internet Draft 02 (Oct. 15, 1999). (Patel).

Goncalves, et al. Check Point FireWall—1 Administration Guide, McGraw–Hill Companies (2000). (Goncalves, Checkpoint FW).

"Building a Microsoft VPN: A Comprehensive Collection of Microsoft Resources," FirstVPN, (Jan. 2000). (FirstVPN Microsoft).

Gulbrandsen, Vixie & Esibov, *A DNS RR for specifying the location of services* (*DNS SRV*), IETF RFC 2782 (Feb. 2000). (RFC 2782, DNS SRV).

Mitre Organization, "Technical Description," Collaborative Operations in Joint Expeditionary Force Experiment (JEFX) 99 (Feb. 2000). (Mitre, SIPRNET).

H. Schulzrinne, et al. "Application–Layer Mobility Using SIP," Mobile Computing and Communications Review, vol. 4, No. 3, pp. 47–57 (Jul. 2000). (Application, SIP).

Kindred et al, "Dynamic VPN Communities: Implementation and Experience," DARPA Information Survivability Conference and Exposition II (Jun. 2001). (DARPA, VPN Systems).

ANX 101: Basic ANX Service Outline. (Outline, ANX).

ANX 201: Advanced ANX Service. (Advanced, ANX).

Appendix A: Certificate Profile for ANX IPsec Certificates. (Appendix, ANX).

Assured Digital Products. (Assured Digital).

Aventail Corp., "Aventail AutoSOCKS the Client Key to Network Security," Aventail Corporation White Paper. (Network Security, Aventail).

Cindy Moran, "DISN Data Networks: Secret Internet Protocol Router Network (SIPRNet)." (Moran, SIPRNET).

Data Fellows F–Secure VPN+ (F–Secure VPN+).

Interim Operational Systems Doctrine for the Remote Access Security Program (RASP) Secret Dial–In Solution. (RASP, SIPRNET).

Onion Routing, "Investigation of Route Selection Algorithms," available at http://www.onion–router.net/Archives/Route/Index.html. (Route Selection, Onion Routing).

Secure Computing, "Buttet–Proofing an Army Net," Washington Technology. (Secure, SIPRNET).

Sparta "Dynamic Virtual Private Network," (Sparta, VPN Systems).

Standard Operation Procedure for Using the 1910 Secure Modems. (Standard, SIPRNET).

Publically available emails relating to FreeS/WAN (MSFTVX00018833–MSFTVX00019206). (FreeS/WAN emails, FreeS/WAN).

Kaufman et al., "implementing IPsec," (Copyright 1999) (Implementing IPsec).

Network Associates *Gauntlet Firewall For Unix User's Guide Version 5.0* (1999). (Gauntlet User's Guide—Unix, Firewall Products).

Network Associates *Gauntlet Firewall For Windows NT Getting Started Guide Version 5.0* (1999) (Gauntlet Getting Started Guide—NT, Firewall Products).

Network Associates *Gauntlet Firewall For Unix Getting Started Guide Version 5.0* (1999) (Gauntlet Unix Getting Started Guide, Firewall Products).

Network Associates *Release Notes Gauntlet Firewall for Unix 5.0* (Mar. 19, 1999) (Gauntlet Unix Release Notes, Firewall Products).

Network Associates *Gauntlet Firewall For Windows NT Administrator's Guide Version 5.0* (1999) (Gauntlet NT Administrator's Guide, Firewall Products).

Trusted Information Systems, Inc. *Gauntlet Internet Firewall Firewall–to–Firewall Encryption Guide Version 3.1* (1996) (Gauntlet Firewall–to–Firewall, Firewall Products).

Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).

Network Associates *Gauntlet Firewall For Unix Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN)

Dan Sterne *Dynamic Virtual Private Networks* (May 23, 2000) (Sterne DVPN, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks* (*DVPN*) (Dec. 21, 1999) (Kindred DVPN, DVPN).

Dan Sterne et al. *TIS Dynamic Security Perimeter Research Project Demonstration* (Mar. 9, 1998) (Dynamic Security Perimeter, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks Capability Description* (Jan. 5, 2000) (Kindred DVPN Capability, DVPN) 11.

Oct. 7, and 28 1997 email from Domenic J. Turchi Jr. (SPARTA00001712–1714, 1808–1811) (Turchi DVPN email, DVPN).

James Just & Dan Sterne *Security Quickstart Task Update* (Feb. 5, 1997) (Security Quickstart, DVPN).

Virtual Private Network Demonstration dated Mar. 21, 1998 (SPARTA00001844–54) (DVPN Demonstration, DVPN).

GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration* (IFD) 1.1 Plan (Mar. 10, 1998) (IFD 1.1, DVPN).

Microsoft Corp. Windows NT Server Product Documentation: Administration Guide—Connection Point Services, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cpsops.mspx (Connection Point Services) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows. Accordingly, upon information and belief, this reference is prior art to the patents–insuit).

Microsoft Corp. Windows NT Servier Product Documentation: Administration Kit Guide—Connection Manager, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cmak.mspx (Connection Manager) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.).

Microsoft Corp. Autodial Heuristics, available at http://support.microsoft.com/kb/164249 (Autodial Heuristics) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.).

Microsoft Corp., Cariplo: Distributed Component Object Model, (1996) available at http://msdn2.microsoft.com/en–us/library/ms809332(printer).aspx (Cariplo I).

Marc Levy, COM Internet Services (Apr. 23, 1999), available at http://msdn2.microsoft.com/en–us/library/ms809302(printer).aspx (Levy).

Markus Horstmann and Mary Kirtland, DCOM Architecture (Jul. 23, 1997), available at http://msdn2.microsoft.com/en–us/library/ms809311(printer).aspx (Horstmann).

Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at http://msdn2.microsoft.com/en–us/library/ms809320(printer).aspx (DCOM Business Overview I).

Microsoft Corp., DCOM Technical Overview (Nov. 1996), available at http://msdn2.microsoft.com/en–us/library/ms809340(printer).aspx (DCOM Technical Overview I).

Microsoft Corp., DCOM Architecture White Paper (1998) available in PDC DVD–ROM (DCOM Architecture).

Microsoft Corp, DCOM—The Distributed Component Object Model, A Business Overview White Paper (Microsoft 1997) available in PDC DVD–ROM (DCOM Business Overview II).

Microsoft Corp., DCOM—Cariplo Home Banking Over The Internet White Paper (Microsoft 1996) available in PDC DVD–ROM (Cariplo II).

Microsoft Corp., DCOM Solutions in Action White Paper (Microsoft 1996) available in PDC DVD–ROM (DCOM Solutions in Action).

Microsoft Corp., DCOM Technical Overview White Paper (Microsoft 1996) available 12 in PDC DVD–ROM (DCOM Technical Overview II).

125. Scott Suhy & Glenn Wood, DNS and Microsoft Windows NT 4.0 (1996) available at http://msdn2.microsoft.com/en–us/library/ms810277(printer).aspx (Suhy).

126. Aaron Skonnard, *Essential Winlnet* 313–423 (Addison Wesley Longman 1998) (Essential Winlnet).

Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at http://msdn2.microsoft.com/enus/library/ms811078 (printer).aspx (Using PPTP).

Microsoft Corp. Internet Connection Services for MS RAS, Standard Edition, http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstart.mspx (Internet Connection Services I).

Microsoft Corp. Internet Connection Services for RAS, Commercial Edition, available athttp://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspx (Internet Connection Services II).

Microsoft Corp., Internet Explorer 5 Corporate Deployment Guide—Appendix B:Enabling Connections with the Connection Manager Administration Kit, available at http://www.microsoft.com/technet/prodtechnol/ie/deploy/deploy5/appendb.mspx (IE5 Corporate Development).

Mark Minasi, *Mastering Windows NT Server 4* 1359–1442 (6th ed., Jan. 15, 1999)(Mastering Windows NT Server).

Hands On, Self–Paced Training for Supporting Verion 4.0 371–473 (Microsoft Press 1998) (Hands On).

Microsoft Corp., MS Point–to–Point Tunneling Protocol (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/maintain/featusability/pptpwp3.mspx (MS PPTP).

Kenneth Gregg, et al., *Microsoft Windows NT Server Adminstrator's Bible* 173–206, 883–911, 974–1076 (IDG Books Worldwide 1999) (Gregg).

Microsoft Corp., Remote Access (Windows), available at http://msdn2.microsoft.com/en–us/library/bb545687 (VS.85,printer).aspx (Remote Access).

Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspx (Understanding PPTP NT 4) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.).

Microsoft Corp., Windows NT 4.0: Virtual Private Networking, available at http://www.microsoft.com/technet/archive/winntas/deploy/confeat/vpntwk.mspx (NT4 VPN) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.).

Anthony Northrup, *NT Network Plumbing: Routers, Proxies, and Web Services* 299–399 (IDG Books Worldwide 1998) (Network Plumbing).

Microsoft Corp., Chapter 1—Introduction to Windows NT Routing with Routing and Remote Access Service, Available at http://www.microsoft.com/technet/archive/winntas/proddocs/ rras40/rrasch01.mspx (Intro to RRAS) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.) 13.

Microsoft Corp., Windows NT Server Product Documentation: Chapter 5—Planning for Large–Scale Configurations, available at http://www.microsoft.com/technet/archive/winntas/proddocs/rras40/rrasch05.mspx (Large–Scale Configurations) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents–in–suit.).

F–Secure, F–Secure Evaluation Kit (May 1999) (FSECURE 00000003) (Evaluation Kit 3).

F–Secure, F–Secure NameSurfer (May 1999) (FSECURE 00000003) (NameSurfer 3).

F–Secure, F–Secure VPN Administrator's Guide (May 1999) (from FSECURE 00000003) (F–Secure VPN 3).

F–Secure, F–Secure SSH User's & Administrator's Guide (May 1999) (from FSECURE 00000003) (SSH Guide 3).

F–Secure, *F–Secure SSH2.0 for Windows NT and 95* (May 1999) (from FSECURE 00000003) (SSH 2.0 Guide 3).

F–Secure, *F–Secure VPN+ Administrator's Guide* (May 1999) (from FSECURE 00000003) (VPN+ Guide 3).

F. Secure, *F–Secure VPN+ 4.1* (1999) (from FSECURE 00000006) (VPN+ 4.1 Guide 6).

F–Secure, *F–Secure SSH* (1996) (from FSECURE 00000006) (F–Secure SSH 6).

F–Secure, *F–Secure SSH 2.0 for Windows NT and 95* (1998) (from FSECURE 00000006) (F–Secure SSH 2.0 Guide 6).

F–Secure, *F–Secure Evaluation Kit* (Sep. 1998) (FSECURE 00000009) (Evaluation Kit 9).

F–Secure, *F–Secure SSH User's & Administrator's Guide* (Sep. 1998) (from FSECURE 00000009) (SSH Guide 9).

F–Secure, *F–Secure SSH 2.0 for Windows NT and 95* (Sep. 1998) (from FSECURE 00000009) (F–secure SSH 2.0 Guide 9).

F–Secure, *F–Secure VPN+* (Sep. 1998) (from FSECURE 00000009) (VPN+ Guide 9).

F–Secure, *F–Secure Management Tools Administrator's Guide* (1999) (from FSECURE 00000003) (F–secure Management Tools).

F–Secure, *F–Secure Desktop, User's Guide* (1997) (from FSECURE 00000009) (F–secure Desktop User's Guide).

SafeNet, Inc., *VPN Policy Manager* (Jan. 2000) (VPN Policy Manager).

F–Secure, *F–Secure VPN+ for Windows NT 4.0* (1998) (from FSECURE 00000009) (F–secure VPN+).

IRE, Inc., *SafeNet/Soft–PK Version 4* (Mar. 28, 2000) (Soft–PK Version 4).

IRE/SafeNet Inc., *VPN Technologies Overview* (Mar. 28, 2000) (Safenet VPN Overview).

IRE, Inc., *SafeNet/Security Center Technical Reference Addendum* (Jun. 22, 1999) (Safenet Addendum).

IRE, Inc., *System Description for VPN Policy Manager and SafeNet/SoftPK* (Mar. 30, 2000) (VPN Policy Manager System Description).

IRE, Inc., About SafeNet/VPN Policy Manager (1999) (About Safenet VPN Policy Manager).

IRE, Inc., *SafeNet/VPN Policy Manager Quick Start Guide Version 1* (1999) (SafeNet VPN Policy Manager).

Trusted Information Systems, Inc., *Gauntlet Internet Firewall, Firewall Product Functional Summary* (Jul. 22, 1996) (Gauntlet Functional Summary).

Trusted Information Systems, Inc., *Running the Gauntlet Internet Firewall, An Administrator's Guide to Gauntlet Version 3.0* (May 31, 1995) (Running the Gauntlet Internet Firewall).

Ted Harwood, *Windows NT Terminal Server and Citrix Metaframe* (New Riders 1999) (Windows NT Harwood) 79.

Todd W. Matehrs and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implemetning Terminal Server and Citrix MetaFrame* (Macmillan Technical Publishing 1999) (Windows NT Mathers).

Bernard Aboba et al., *Securing L2TP using IPSEC* (Feb. 2, 1999).

156. *Finding Your Way Through the VPN Maze* (1999) ("PGP").

Linux FreeS/WAN Overview (1999) (Linux FreeS/WAN) Overview).

TimeStep, *The Business Case for Secure VPNs* (1998) ("TimeStep").

WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).

WatchGuard Technologies, Inc., *MSS Firewall Specifications* (1999).

WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).

WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).

WatchGuard Technologies, Inc., *WatchGuard LiveSecurity for MSS Powerpoint* (Feb. 14, 2000).

WatchGuard Technologies, Inc., *MSS Version 2.5, Add–On for WatchGuard SOHO Releaset Notes* (Jul. 21, 2000).

Air Force Research Laboratory, *Statement of Work for Information Assurance System Architecture and Integration, PR No. N–8–6106* (Contract No. F30602–98–C–0012) (Jan. 29, 1998).

GTE Internetworking & BBN Technologies DARPA *Information Assurance Program Integrated Feasibility Demonstration (IFD)* 1.2 Report, Rev. 1.0 (Sep. 21, 1998).

BBN Information Assurance Contract, *TIS Labs Monthly Status Report* (Mar. 16–Apr. 30, 1998).

DARPA, *Dynamic Virtual Private Network (VPN) Powerpoint.*

GTE Internetworking, *Contractor's Program Progress Report* (Mar. 16–Apr. 30, 1998).

Darrell Kindred, *Dynamic Virtual Private Networks (DVPN) Countermeasure Characterization* (Jan. 30, 2001).

Virtual Private Networking Countermeasure Characterization (Mar. 30, 2000).

Virtual Private Network Demonstration (Mar. 21, 1998).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks (VPNs) and Integrated Security Management* (2000).

Information Assurance/NAI Labs, *Create/Add DVPN Enclave*(2000).

NAI Labs, *IFE 3.1 Integration Demo* (2000).

Information Assurance, *Science Fair Agenda* (2000).

Darrell Kindred et al., *Proposed Threads for IFE 3.1* (Jan. 13, 2000).

IFE 3.1 Technology Dependencies (2000).

IFE 3.1 Topology (Feb. 9, 2000).

Information Assurance, *Informatiion Assurance Integration: IFE 3.1, Hypothesis & Thread Development* (Jan. 10–11, 2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation* (2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.2* (2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).

T. Braun et al., *Virtual Private Network Architecture,* Charging and Accounting Technology for the Internet (Aug. 1, 1999) (VPNA).

Network Associates Products—*PGP Total Network Security Suite, Dynamic Virtual Private Networks* (1999).

Microsoft Corporation, Microsoft Proxy Server 2.0 (1997) (Proxy Server 2.0, Microsoft Prior Art VPN Technology).

David Johnson et al., *A Guide To Microsoft Proxy Server 2.0* (1999) (Johnson, Microsoft Prior Art VPN Technology).

Microsoft Corporation, *Setting Server Parameters* (1997 (Proxy Server 2.0 CD labeled MSFTVX00157288) (Setting Server Parameters, Microsoft Prior Art VPN Technology).

Kevin Schuler, *Microsoft Proxy Server 2* (1998) (Schuler, Microsoft Prior Art VPN Technology).

Erik Rozell et al., *MCSE Proxy Server 2 Study Guide* (1998) (Rozell, Microsoft Prior 15 Art VPN Technology.

M. Shane Stigler & Mark A. Linsenbardt, *IIS 4 and Proxy Server 2* (1999) (Stigler, Microsoft Prior Art VPN Technology).

David G. Schaer, *MCSE Test Success: Proxy Server 2* (1998) (Schaer, Microsoft Prior Art VPN Technology).

John Savill, *The Windows NT and Windows 2000 Answer Book* (1999) (Savill, Microsoft Prior Art VPN Technology).

Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).

Network Associates *Gauntlet Firewall For UNIX Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).

File History for U.S. Appl. No. 09/653,201, Applicant(s): Whittle Bryan, et al., filed Aug. 31, 2000.

AutoSOCKS v2.1, Datasheet, http://web.archive.org/web/19970212013409/www.aventail.com/prod/autoskds.html.

Ran Atkinson, *Use of DNS to Distribute Keys,* Sep. 7, 1993, http://ops.ietf.org/lists/namedroppers/namedroppers.199x/msg00945.html.

FirstVPN Enterprise Networks, Overview.

Chapter 1: Introduction to Firewall Technology, Administration Guide: Dec. 19, 2007, http://www.books24x7.com/book/id_762/viewer_r.asp?bookid=762&chunked=41065062.

The TLS Protocol Version 1.0; Jan. 1999; p. 65 of 71.

Elizabeth D. Zwicky, et al., Building Internet Firewalls, 2nd Ed.

Virtual Private Networks—Assured Digital Incorporated—ADI 4500; http://web.archive.org/web/19990224050035/www.assured–digital.com/products/prodvpn/adia4500.htm.

Accessware—The Third Wave in Network Security, Conclave from Internet Dynamics; http://web.archive.org/web/11980210013830/interdyn.com/Accessware.html.

Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks,* www.extendedsystems.com.

Socks Version 5; Executive Summary; http://web.archive.org/web/19970620031945/www.aventail.com/educate/whitepaper/sockswp.html.

Internet Dynamics First to Ship Integrated Security Solutions for Enterprise Intranets and Extranets; Sep. 15, 1997; http://web.archive.org/web/19980210014150/interdyn.com.

E-mails from various individuals to Linux IPsec re:DNS-LDAP Splicing.

Microsoft Corporation's Fifth Amended Invalidity Contentions dated Sep. 18, 2009, *VirnetX Inc. and Science Applications International Corp.* v. *Microsoft Corporation* and invalidity claim charts for U.S. Patent Nos. 7,188,180 and 6,839,759.

The IPSEC Protocol as described in Atkinson, et al., "Security Architecture for the Internet Protocol," Networking Working Group, RFC 2401 (Nov. 1998) ("RFC 2401"); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and R. Glenn, "The Use HMAC-SHA-1-96 with ESP and AH," RFC 2404 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

S.Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Derrell Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Douglas Maughan, et al, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

D. Harkins and D. Carrell, "The Internet Key Exchange (IKE)," RFC 2409 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec," RFC 2410 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

R. Thayer, et al., "IP Security Document Roadmap," RFC 2411 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Hilarie K. Orman, "The Oakley Key Determination Protocol," RFC 2412 (Nov. 1998) in combination with J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose California (Jul. 1996) ("Galvin").

David Kosiur, "Building and Managing Virtual Private Networks" (1998).

P. Mockapetris, "Domain Names—Implementation and Specification," Network Working Group, RFC 1035 (Nov. 1987).

Request for Inter Partes Reexamination of Patent No. 7,188,180, dated Nov. 25, 2009.

Exhibit 2 "Aventail Connect v3.1/v2.6 Administrator's Guide", 120 pages, 1996-1999.

Exhibit 3A, "Gauntlet Firewall for Windows", pp. 1-137, 1998-1999.

Exhibit 3B, "Gauntlet Firewall for Windows", pp. 138-275, 1998-1999.

Exhibit 4, "Kosiur", Building and Managing VPNs, pp. 1-396, 1998.

Exhibit 5, Building a Microsoft VPN; A comprehensive Collection of Microfoft Resources, pp. 1-216.

Exhibit 6, Windows NT Server, Virtual Private Network; An Overview, pp. 1-26, 1998.

Exhibit 7, "Networking Working Group Request for Comments: 1035" pp. 1-56, 1987.

INTER PARTES REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 316

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **1-10** and **12** is confirmed.

New claim **18** is added and determined to be patentable.

Claims **11** and **13-17** were not reexamined.

18. A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:

(1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;

(2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and

(3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer, wherein:

steps (2) and (3) are performed at a DNS server separate from the client computer, and step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to resolve addresses of non secure target computers and, if not so authorized, returning an error from the DNS request.

* * * * *

TAB 7

U 7288569

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

March 30, 2011

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *7,418,504*
ISSUE DATE: *August 26, 2008*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. WALLACE
Certifying Officer

UNITED STATES PATENT AND TRADEMARK OFFICE
DEPARTMENT OF COMMERCE

Plaintiffs' VirnetX Exhibit
VirnetX, Inc. v. Apple, Inc.
PX004
C.A. 6:10-cv-0417

US007418504B2

(12) **United States Patent**
Larson et al.

(10) **Patent No.: US 7,418,504 B2**
(45) **Date of Patent: Aug. 26, 2008**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES**

(75) Inventors: **Victor Larson**, Fairfax, VA (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Edmund Colby Munger**, Crownsville, MD (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **VirnetX, Inc.**, Scotts Valley, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 646 days.

(21) Appl. No.: **10/714,849**

(22) Filed: **Nov. 18, 2003**

(65) **Prior Publication Data**

US 2004/0098485 A1 May 20, 2004

Related U.S. Application Data

(63) Continuation of application No. 09/558,210, filed on Apr. 26, 2000, now abandoned, which is a continuation-in-part of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/137,704, filed on Jun. 7, 1999, provisional application No. 60/106,261, filed on Oct. 30, 1998.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/226**

(58) **Field of Classification Search** 709/226, 709/221; 713/201
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,933,846	A	6/1990	Humphrey et al.
4,988,990	A	1/1991	Warrior
5,164,988	A	11/1992	Matyas et al.
5,276,735	A	1/1994	Boebert et al.
5,311,593	A	5/1994	Carmi

(Continued)

FOREIGN PATENT DOCUMENTS

DE	199 24 575	12/1999

(Continued)

OTHER PUBLICATIONS

Laurie Wells (Lancasterbibelmail MSN Com); "Subject: Security Icon" Usenet Newsgroup, Oct. 19, 1998, XP002200606.

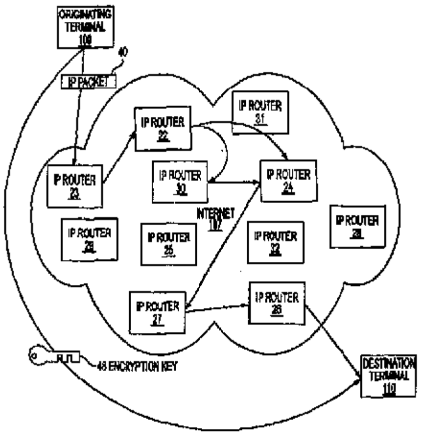
(Continued)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—McDermott Will & Emery, LLP

(57) **ABSTRACT**

A secure domain name service for a computer network is disclosed that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. The portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

60 Claims, 40 Drawing Sheets



U.S. PATENT DOCUMENTS

5,329,521	A	7/1994	Walsh et al.	
5,341,426	A	8/1994	Barney et al.	
5,367,643	A	11/1994	Chang et al.	
5,559,883	A	9/1996	Williams	
5,561,669	A	10/1996	Lenney et al.	
5,588,060	A	12/1996	Aziz	
5,625,626	A	4/1997	Umekita	
5,654,695	A	8/1997	Olnowich et al.	
5,682,480	A	10/1997	Nakagawa	
5,689,566	A	11/1997	Nguyen	
5,740,375	A	4/1998	Dunne et al.	
5,774,660	A	6/1998	Brendel et al.	
5,787,172	A	7/1998	Arnold	
5,790,548	A	8/1998	Sistanizadeh et al.	
5,796,942	A	8/1998	Esbensen	
5,805,801	A	9/1998	Holloway et al.	
5,842,040	A	11/1998	Hughes et al.	
5,845,091	A	12/1998	Dunne et al.	
5,867,650	A	2/1999	Osterman	
5,870,610	A	2/1999	Beyda et al.	
5,878,231	A	3/1999	Baehr et al.	
5,892,903	A	4/1999	Klaus	
5,898,830	A	4/1999	Wesinger, Jr. et al.	
5,905,859	A	5/1999	Holloway et al.	
5,918,019	A	6/1999	Valencia	
5,996,016	A	11/1999	Thalheimer et al.	
6,006,259	A	12/1999	Adelman et al.	
6,006,272	A	12/1999	Aravamudan et al.	
6,016,318	A	1/2000	Tomoike	
6,016,512	A	1/2000	Huitema	
6,041,342	A	3/2000	Yamaguchi	
6,052,788	A	4/2000	Wesinger, Jr. et al.	
6,055,574	A	4/2000	Smorodinsky et al.	
6,061,736	A	5/2000	Rochberger et al.	
6,079,020	A	6/2000	Liu	
6,092,200	A	7/2000	Muniyappa et al.	
6,101,182	A	8/2000	Sistanizadeh et al.	
6,119,171	A	9/2000	Alkhatib	
6,119,234	A	9/2000	Aziz et al.	
6,147,976	A	11/2000	Shand et al.	
6,157,957	A	12/2000	Berthaud	
6,158,011	A	12/2000	Chen et al.	
6,168,409	B1	1/2001	Fare	
6,175,867	B1	1/2001	Taghadoss	
6,178,409	B1	1/2001	Weber et al.	
6,178,505	B1	1/2001	Schneider et al.	
6,179,102	B1	1/2001	Weber et al.	
6,222,842	B1	4/2001	Sasyan et al.	
6,226,751	B1	5/2001	Arrow et al.	
6,233,618	B1	5/2001	Shannon	
6,243,360	B1	6/2001	Basilico	
6,243,749	B1	6/2001	Sitaraman et al.	
6,243,754	B1	6/2001	Guerin et al.	
6,256,671	B1	7/2001	Strentzsch et al.	
6,263,445	B1	7/2001	Blumenau	
6,286,047	B1	9/2001	Ramanathan et al.	
6,301,223	B1	10/2001	Hrastar et al.	
6,308,274	B1	10/2001	Swift	
6,311,207	B1	10/2001	Mighdoll et al.	
6,324,161	B1	11/2001	Kirch	
6,330,562	B1	12/2001	Boden et al.	
6,332,158	B1	12/2001	Risley et al.	
6,353,614	B1	3/2002	Borella et al.	
6,425,003	B1	7/2002	Herzog et al.	
6,430,155	B1	8/2002	Davie et al.	
6,430,610	B1	8/2002	Carter	
6,487,598	B1	11/2002	Valencia	
6,502,135	B1	12/2002	Munger et al.	
6,505,232	B1	1/2003	Mighdoll et al.	
6,510,154	B1	1/2003	Mayes et al.	
6,549,516	B1	4/2003	Albert et al.	
6,557,037	B1	4/2003	Provino	709/227
6,571,296	B1	5/2003	Dillon	
6,571,338	B1	5/2003	Shaio et al.	
6,581,166	B1	6/2003	Hirst et al.	
6,606,708	B1	8/2003	Devine et al.	
6,618,761	B2	9/2003	Munger et al.	
6,671,702	B2	12/2003	Kruglikov et al.	
6,687,551	B2	2/2004	Steindl	
6,714,970	B1	3/2004	Fiveash et al.	
6,717,949	B1	4/2004	Boden et al.	
6,751,738	B2	6/2004	Wesinger, Jr. et al.	
6,760,766	B1	7/2004	Sahlqvist	
6,826,616	B2	11/2004	Larson et al.	
6,839,759	B2	1/2005	Larson et al.	
7,010,604	B1	3/2006	Munger et al.	
7,133,930	B2	11/2006	Munger et al.	
7,188,180	B2	3/2007	Larson et al.	
7,197,563	B2	3/2007	Sheymov et al.	
2002/0004898	A1	1/2002	Droge	
2003/0196122	A1	10/2003	Wesinger, Jr. et al.	
2005/0055306	A1	3/2005	Miller et al.	
2006/0059337	A1	3/2006	Polyhonen et al.	

FOREIGN PATENT DOCUMENTS

DE	199 24 575	A1	12/1999
EP	0 814 589		12/1997
EP	0 814 589	A	12/1997
EP	0 838 930		4/1998
EP	0 838 930	A	4/1998
EP	0 838 930	A2	4/1998
EP	836306	A1	4/1998
EP	0 858 189		8/1998
GB	2 317 792		4/1998
GB	2 317 792	A	4/1998
GB	2 334 181	A	8/1999
WO	9827783	A	6/1998
WO	WO 98/27783		6/1998
WO	WO 98 55930		12/1998
WO	WO 98 59470		12/1998
WO	WO 99 38081		7/1999
WO	WO 99 48303		9/1999
WO	WO 00/17775		3/2000
WO	WO 00/70458		11/2000
WO	WO 01 50688		7/2001

OTHER PUBLICATIONS

Davila J et al, "Implementatin of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf>(Abstract).

Donald E. Eastlake, III, "Domain Name System Security Extensions", Internet Draft, Apr. 1998.

P. Srisuresh, et al., "DNS Extensions to Network Address Translators", Internet Draft, Jul. 1998.

D.B. Chapman, et al., "Building Internet Firewalls, chapters 8 and 10 (parts)", pp. 278-296 and pp. 351-375.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group. Apr. 1998, 51 pages.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-297 and pp. 351-375.

P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.

Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.

W. Stallings, "Cryptography And Network Security", 2nd Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.

Fasbender,Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.

Linux FreeS/WAN Index File, printed from http://liberty.freeswan.org/freeswan_ trees/freeswan-1.3/doc/ on Feb. 21, 2002, 3 Pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_ trees/freeswan-1.3/doc/rationale.html on Feb. 21, 2002, 4 pages.

Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan_ trees/freeswan-1.3/doc/glossary.html on Feb. 21, 2002, 25 pages.

Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ss13/draft302.txt on Feb. 4, 2002, 56 pages.

Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028-1036.

Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.

James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, Aug. 1, 1998, pp. 22-25.

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.

Rich Winkel, "CAQ: Networkinig With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Search Report (dated Oct. 7, 2002), International Application No. PCT/US01/13261.

F. Halsall, "Data Communications, Computer Networks And Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.

Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), "Crowds: Anonymity for Web Transmissoins", pp. 1-23.

Dolev, Shlomi and Ostrovsky, Rafil, "Efficient Anonymous Multicast and Reception"(Extended Abstract), 16 pages.

Rubin, Aviel D., Greer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security" Protection of Location Information in Mobile IP, IEEE publication, 1996, pp. 963-967.

Eastlake, D. E., "Domain Name System Security Extensions", Internet Draft, Apr. 1998, XP002199931, Sections 1, 2.3 and 2.4.

RFC 2401 (dated Nov. 1998) Security Architecture for the Internet Protocol (RTP).

RFC 2543-SIP (dated Mar. 1999): Session Initiation Protocol (SIP or SIPS).

Search Report, IPER (dataed Nov. 13, 2002), International Application No. PCT/US01/04340.

Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.

Shankur, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY, NY 1986.

W. Stallings, "Crytography and Network Security", 2nd, Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

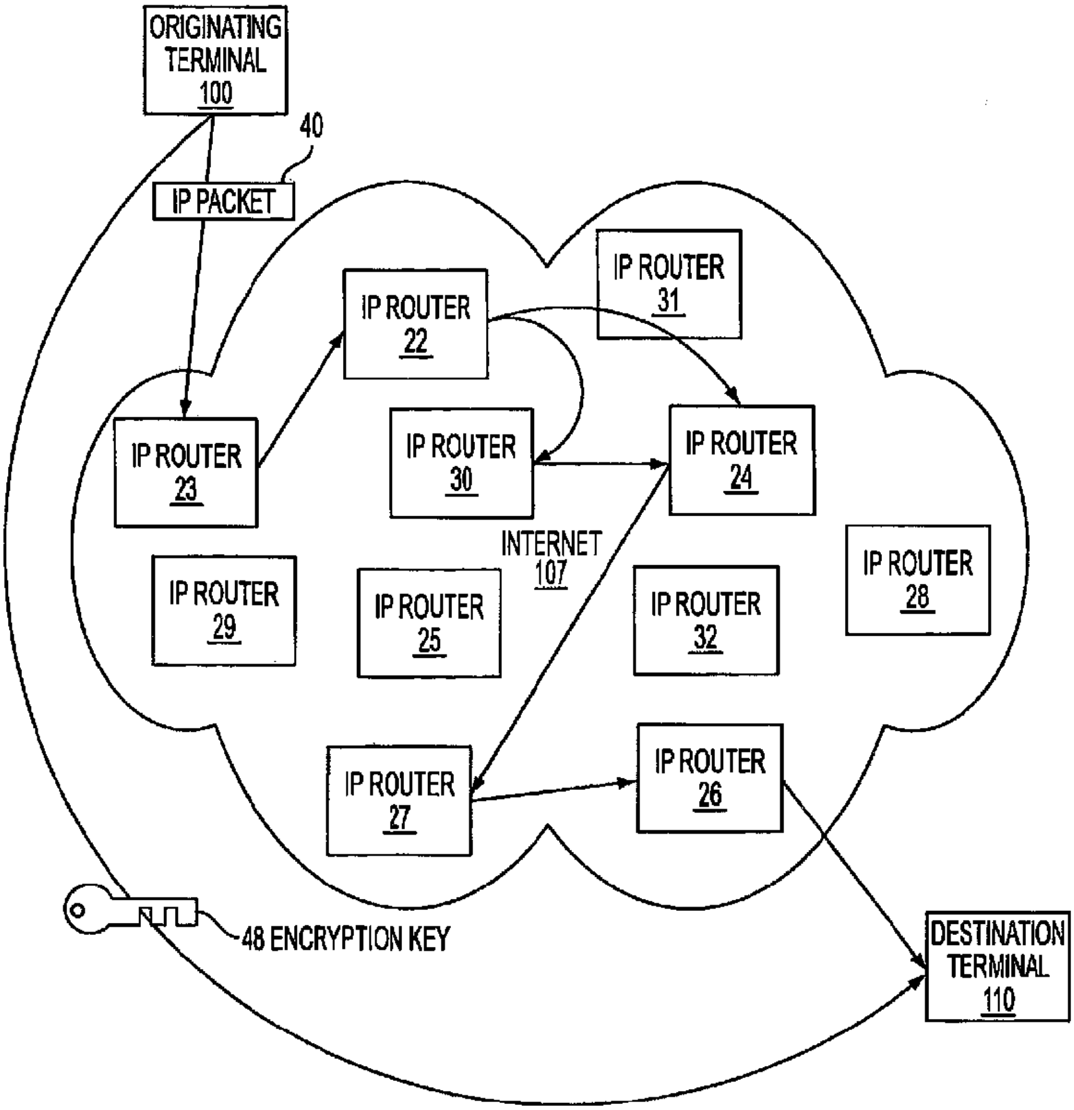

ORIGINATING TERMINAL 100
40
IP PACKET
IP ROUTER 31
IP ROUTER 22
IP ROUTER 23
IP ROUTER 30
IP ROUTER 24
INTERNET 107
IP ROUTER 28
IP ROUTER 29
IP ROUTER 25
IP ROUTER 32
IP ROUTER 27
IP ROUTER 26
48 ENCRYPTION KEY
DESTINATION TERMINAL 110

FIG. 1

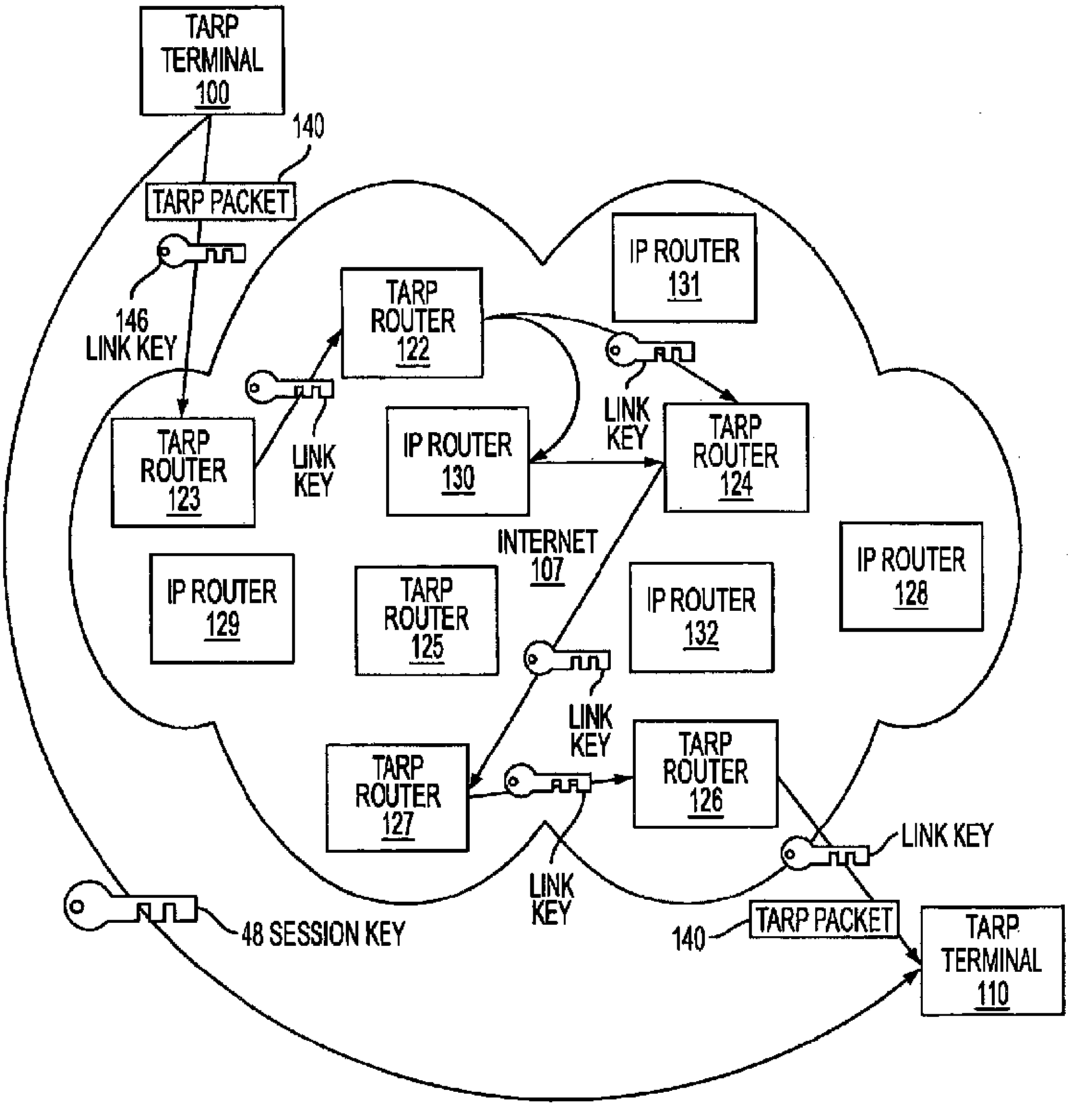

TARP TERMINAL 100
140
TARP PACKET
146 LINK KEY
TARP ROUTER 122
IP ROUTER 131
TARP ROUTER 123
LINK KEY
IP ROUTER 130
LINK KEY
TARP ROUTER 124
INTERNET 107
IP ROUTER 129
TARP ROUTER 125
IP ROUTER 132
IP ROUTER 128
LINK KEY
TARP ROUTER 127
TARP ROUTER 126
LINK KEY
LINK KEY
48 SESSION KEY
140
TARP PACKET
TARP TERMINAL 110

FIG. 2

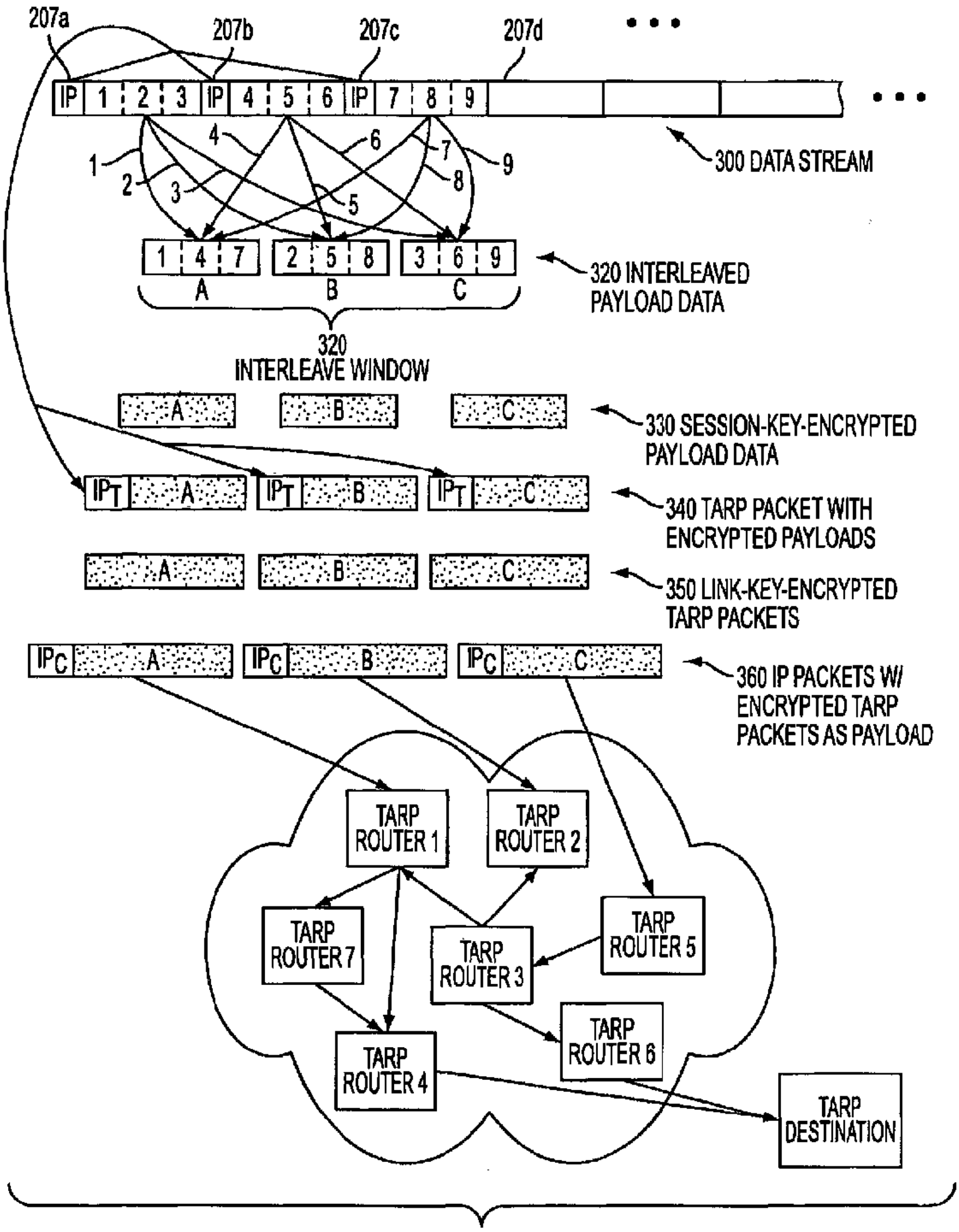

207a
207b
207c
207d
IP
1
2
3
IP
4
5
6
IP
7
8
9
300 DATA STREAM
1
2
3
4
5
6
7
8
9
1 4 7
A
2 5 8
B
3 6 9
C
320 INTERLEAVED PAYLOAD DATA
320 INTERLEAVE WINDOW
A
B
C
330 SESSION-KEY-ENCRYPTED PAYLOAD DATA
IP_T A
IP_T B
IP_T C
340 TARP PACKET WITH ENCRYPTED PAYLOADS
A
B
C
350 LINK-KEY-ENCRYPTED TARP PACKETS
IP_C A
IP_C B
IP_C C
360 IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 4
TARP ROUTER 6
TARP DESTINATION

FIG. 3A

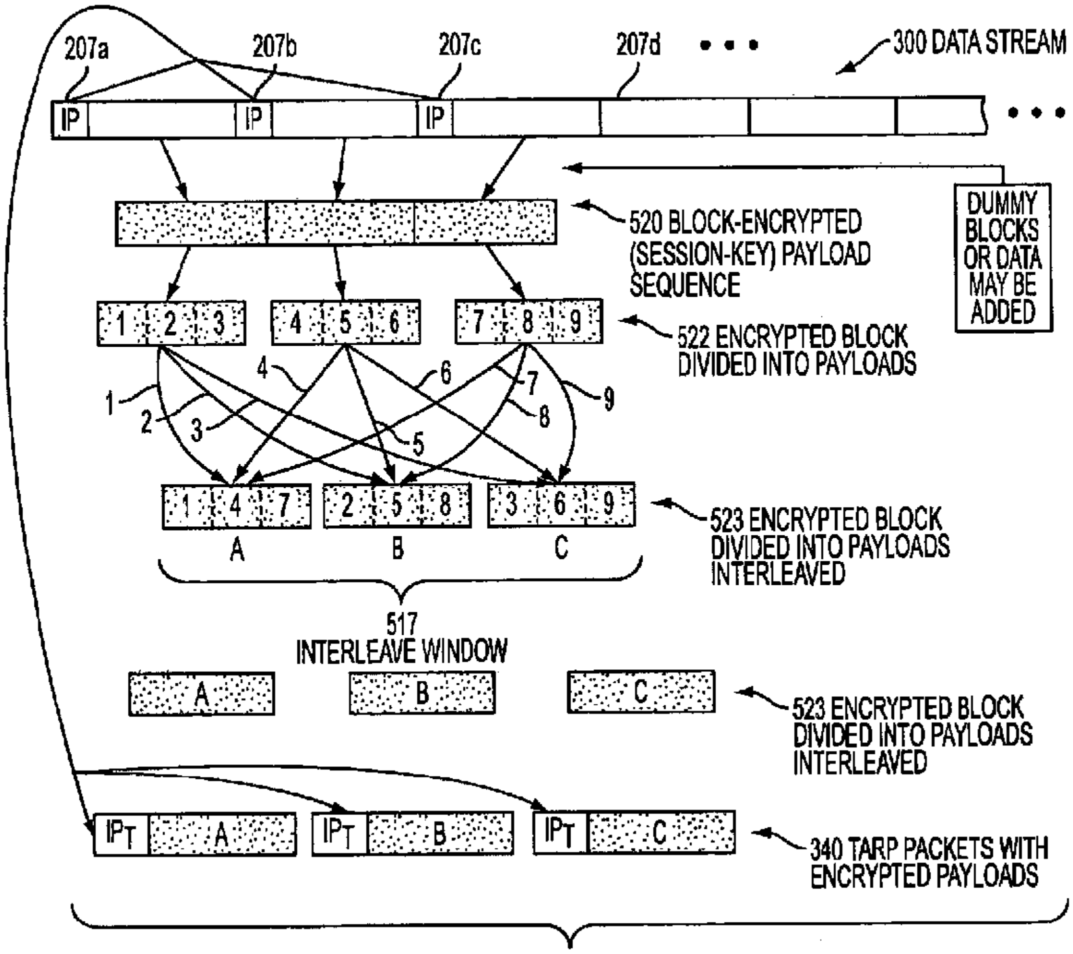

207a
207b
207c
207d
300 DATA STREAM
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
520 BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE
1 2 3
4 5 6
7 8 9
522 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS
1
2
3
4
5
6
7
8
9
1 4 7
2 5 8
3 6 9
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
517 INTERLEAVE WINDOW
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
IPT A
IPT B
IPT C
340 TARP PACKETS WITH ENCRYPTED PAYLOADS

FIG. 3B

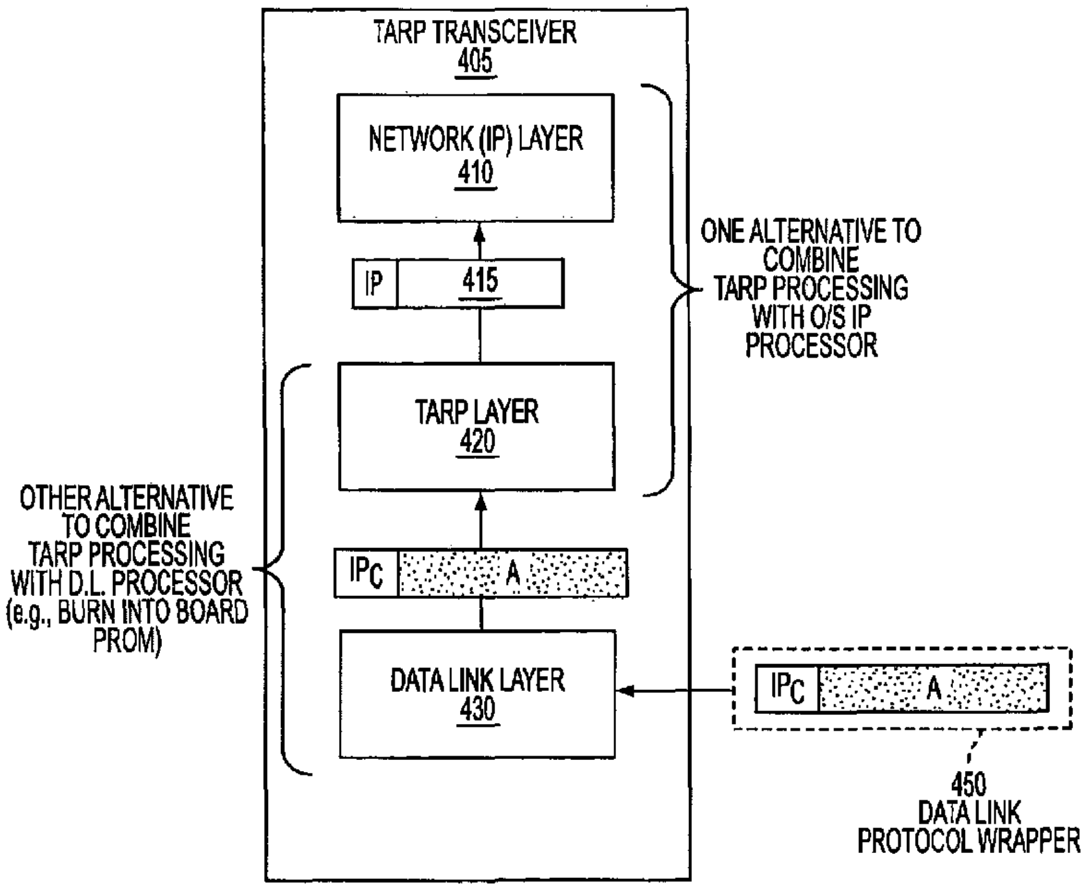

TARP TRANSCEIVER
405
NETWORK (IP) LAYER
410
IP
415
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
TARP LAYER
420
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (e.g., BURN INTO BOARD PROM)
IPC
A
DATA LINK LAYER
430
IPC
A
450
DATA LINK PROTOCOL WRAPPER

FIG. 4

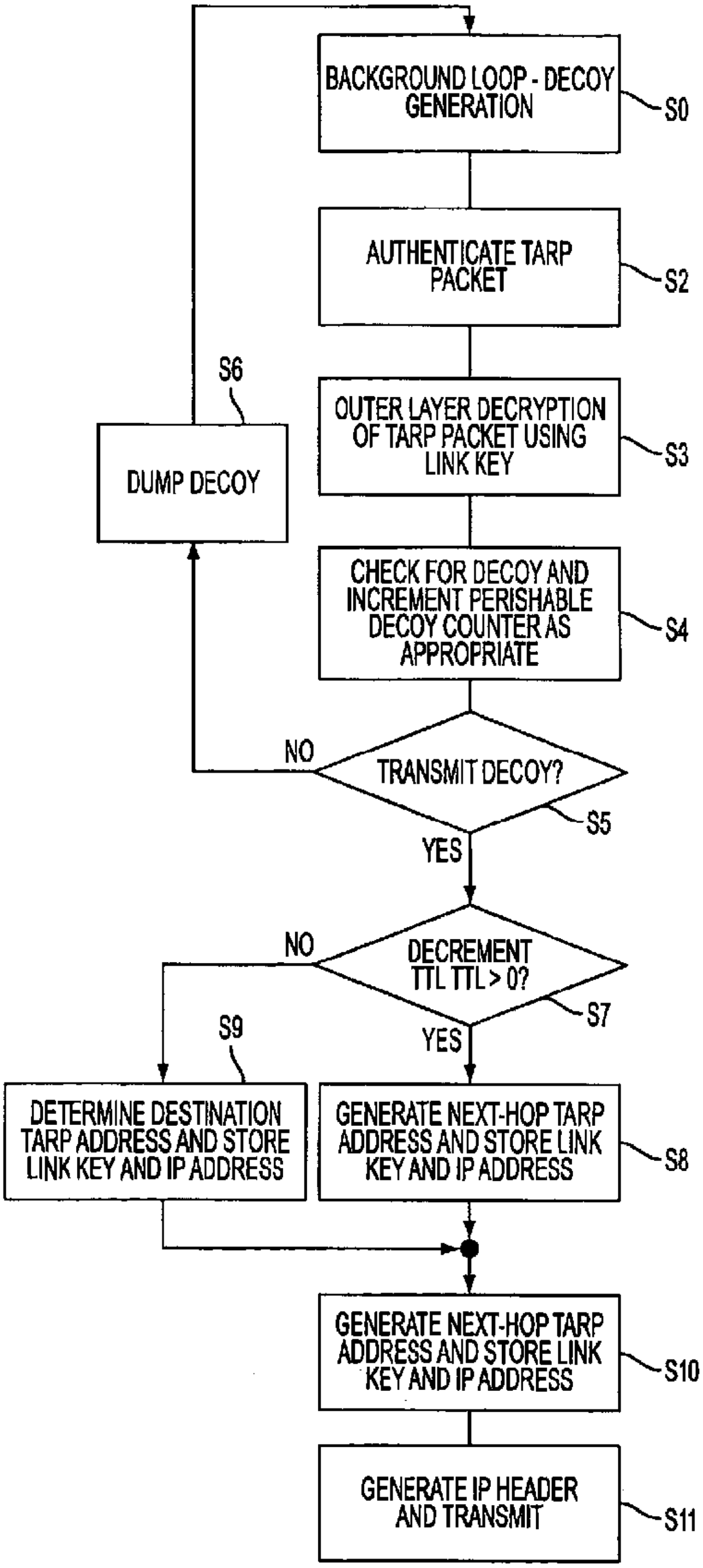

BACKGROUND LOOP - DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
S6
DUMP DECOY
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
S9
YES
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

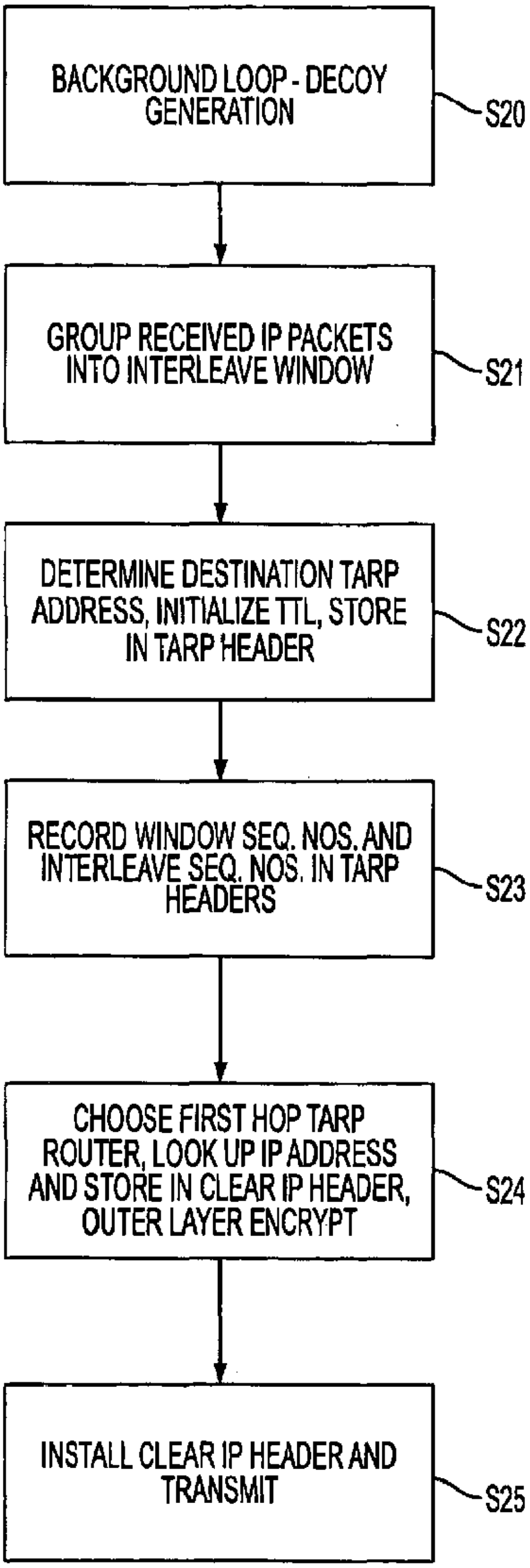

BACKGROUND LOOP - DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS. IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

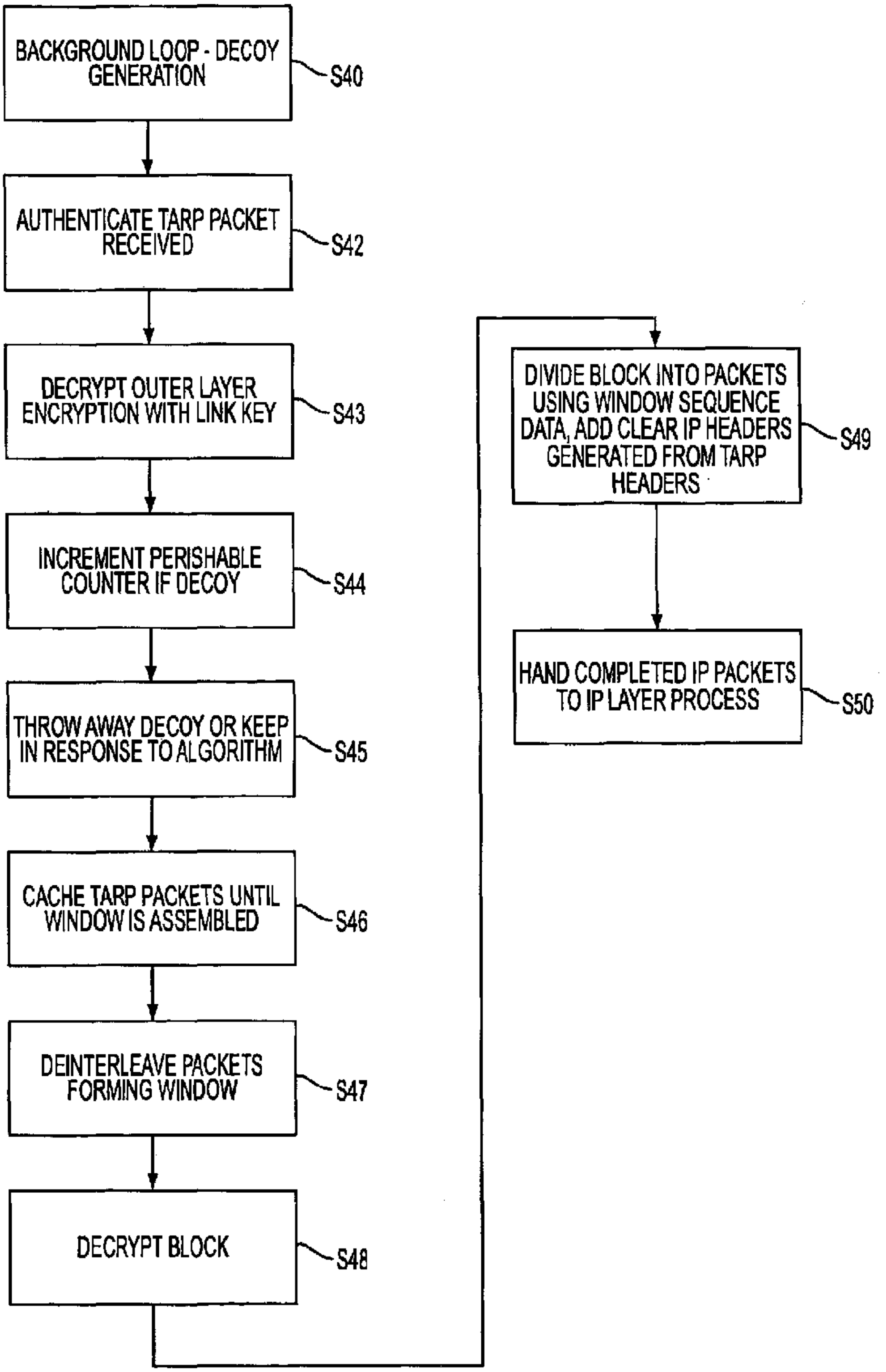

BACKGROUND LOOP - DECOY GENERATION
S40
AUTHENTICATE TARP PACKET RECEIVED
S42
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S43
INCREMENT PERISHABLE COUNTER IF DECOY
S44
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S45
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S46
DEINTERLEAVE PACKETS FORMING WINDOW
S47
DECRYPT BLOCK
S48
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S49
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS
S50

FIG. 7

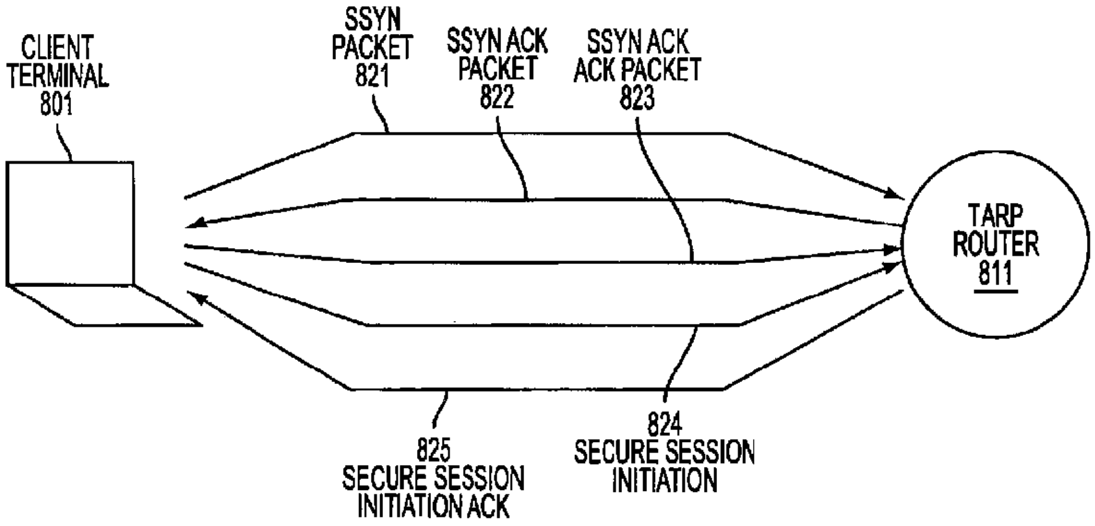
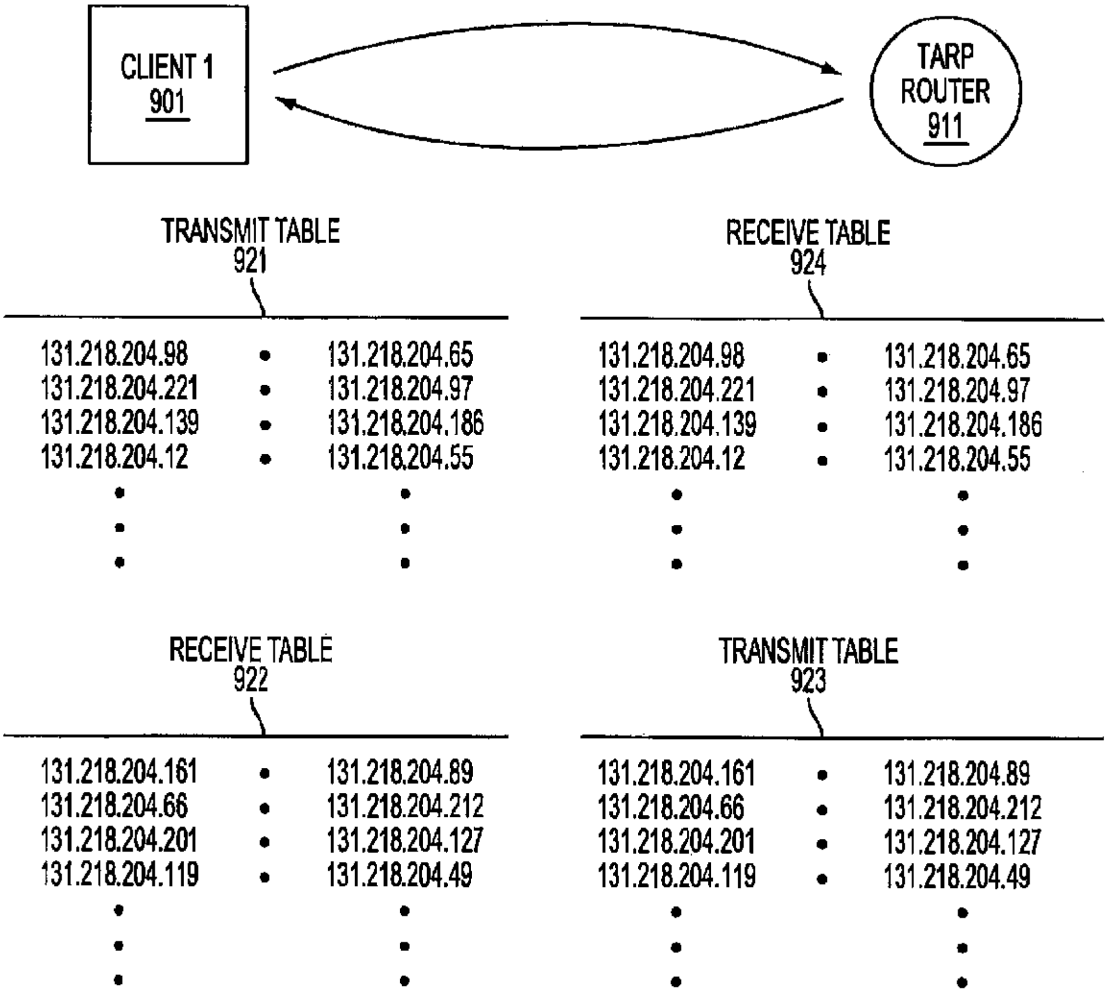

CLIENT
TERMINAL
801
SSYN
PACKET
821
SSYN ACK
PACKET
822
SSYN ACK
ACK PACKET
823
TARP
ROUTER
811
825
SECURE SESSION
INITIATION ACK
824
SECURE SESSION
INITIATION

FIG. 8



TRANSMIT TABLE 921

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
•		•
•		•
•		•

RECEIVE TABLE 924

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
•		•
•		•
•		•

RECEIVE TABLE 922

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
•		•
•		•
•		•

TRANSMIT TABLE 923

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
•		•
•		•
•		•

FIG. 9

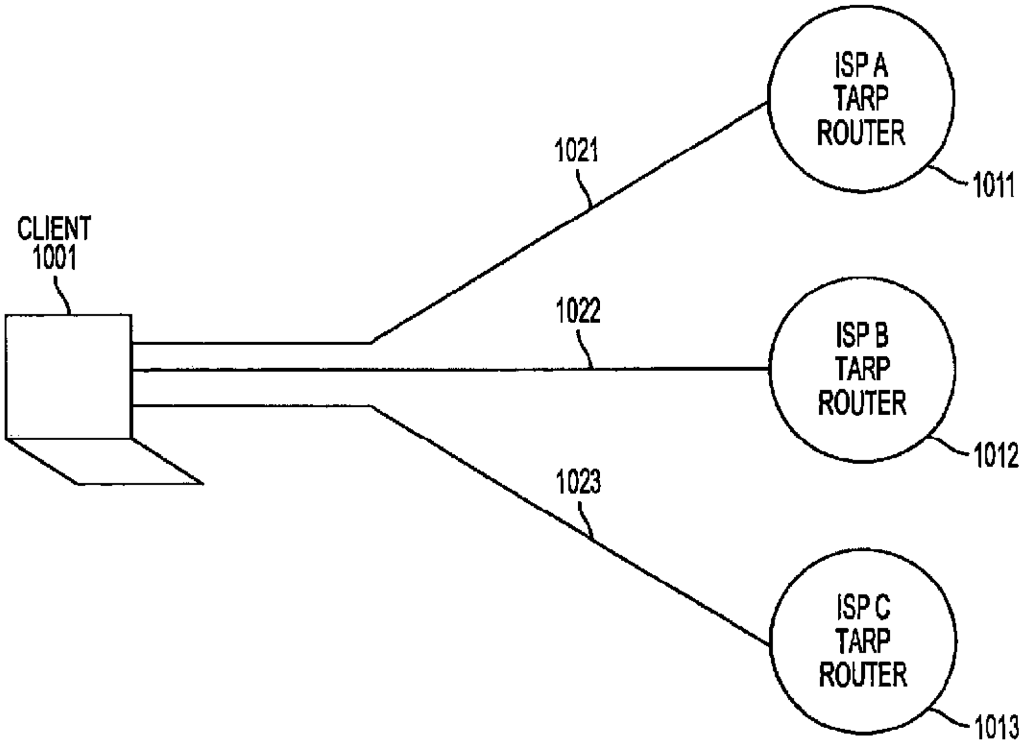

CLIENT
1001
1021
1022
1023
ISP A
TARP
ROUTER
1011
ISP B
TARP
ROUTER
1012
ISP C
TARP
ROUTER
1013

FIG. 10

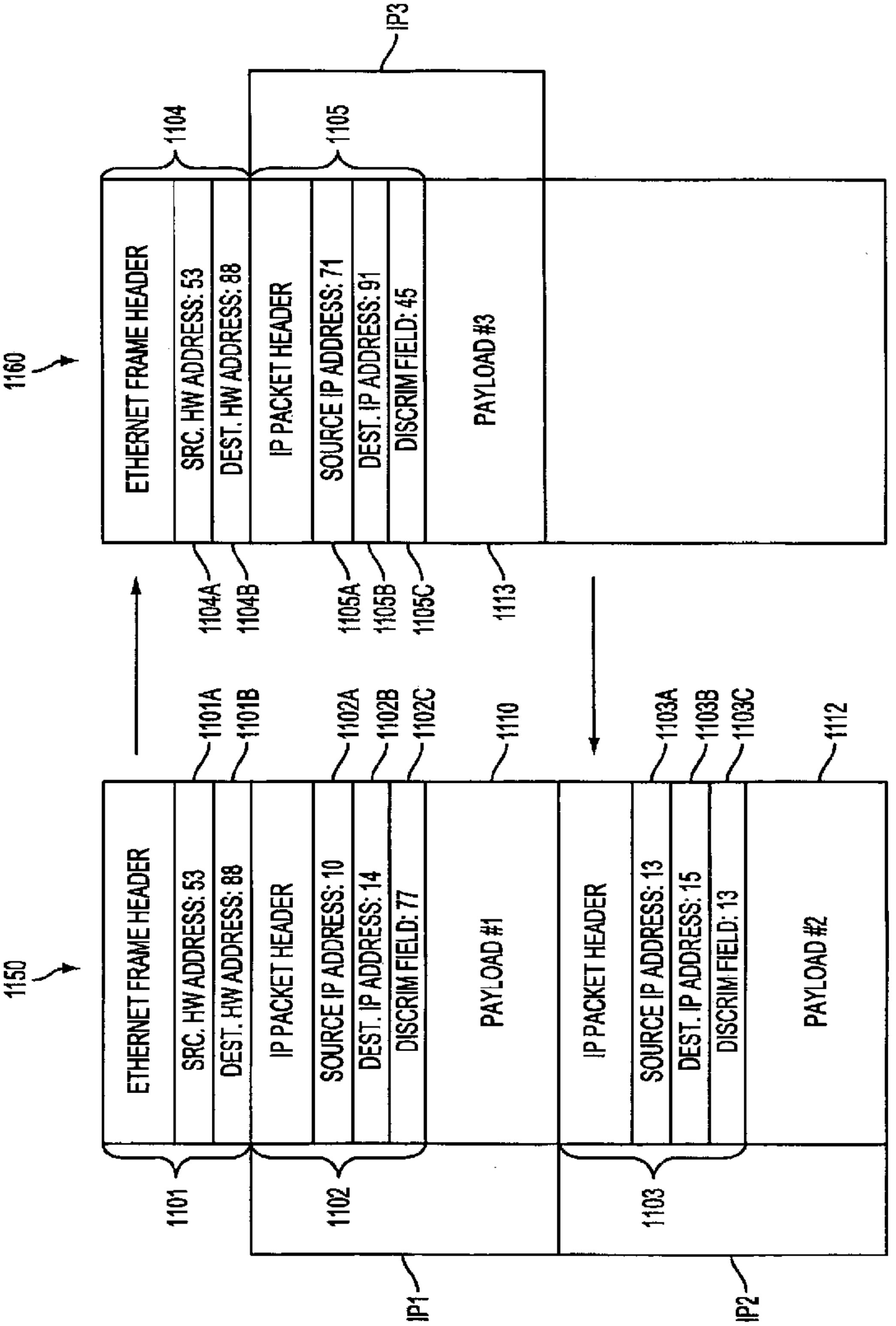
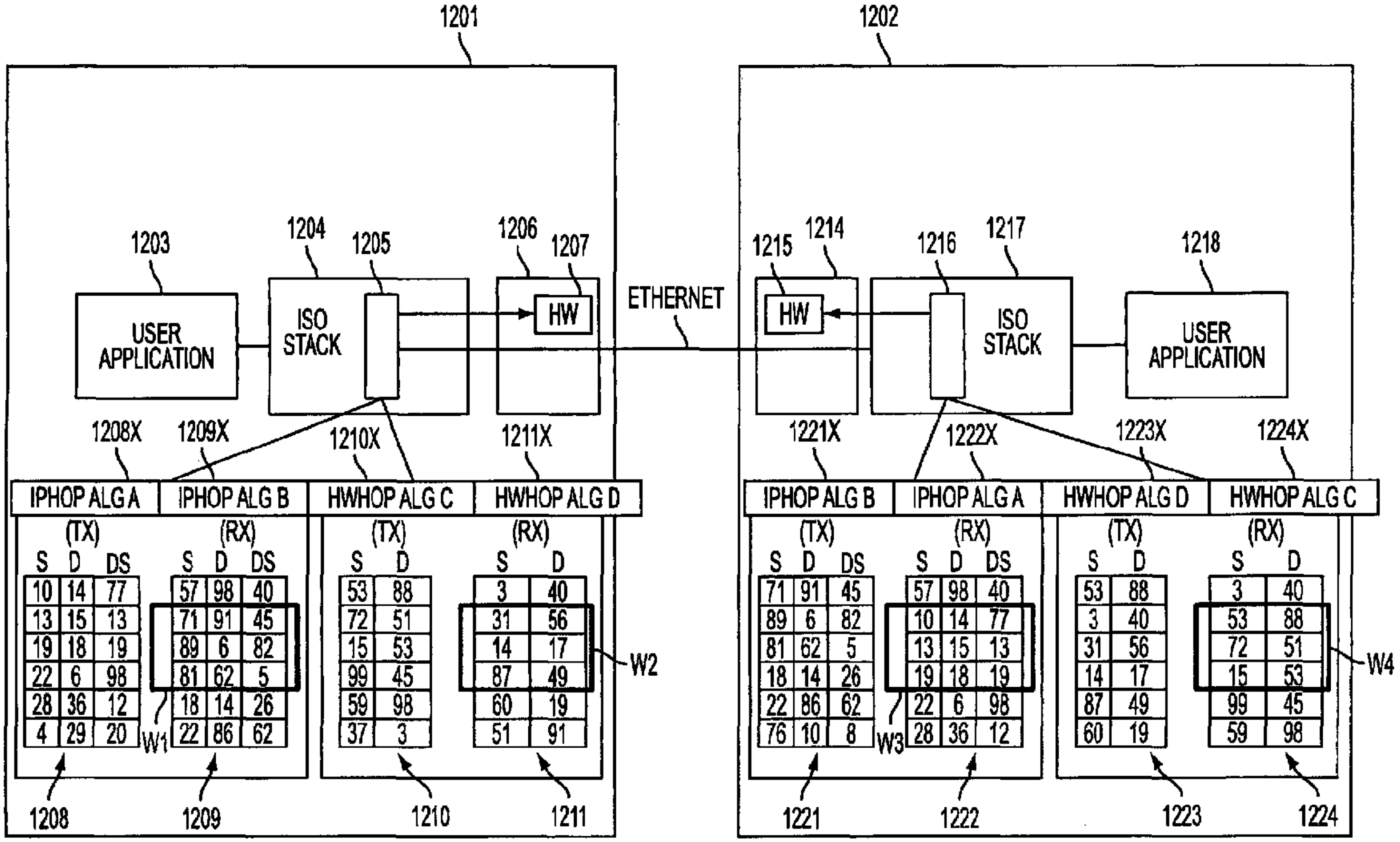

1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104
1105
IP3
1104A
1104B
1105A
1105B
1105C
1113
1150
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1101
1102
1103
IP1
IP2

FIG. 11



1208 IPHOP ALG A (TX)

S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

1209 IPHOP ALG B (RX)

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

1210 HWHOP ALG C (TX)

S	D
53	88
72	51
15	53
99	45
59	98
37	3

1211 HWHOP ALG D (RX)

S	D
3	40
31	56
14	17
87	49
60	19
51	91

1221 IPHOP ALG B (TX)

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

1222 IPHOP ALG A (RX)

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

1223 HWHOP ALG D (TX)

S	D
53	88
3	40
31	56
14	17
87	49
60	19

1224 HWHOP ALG C (RX)

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

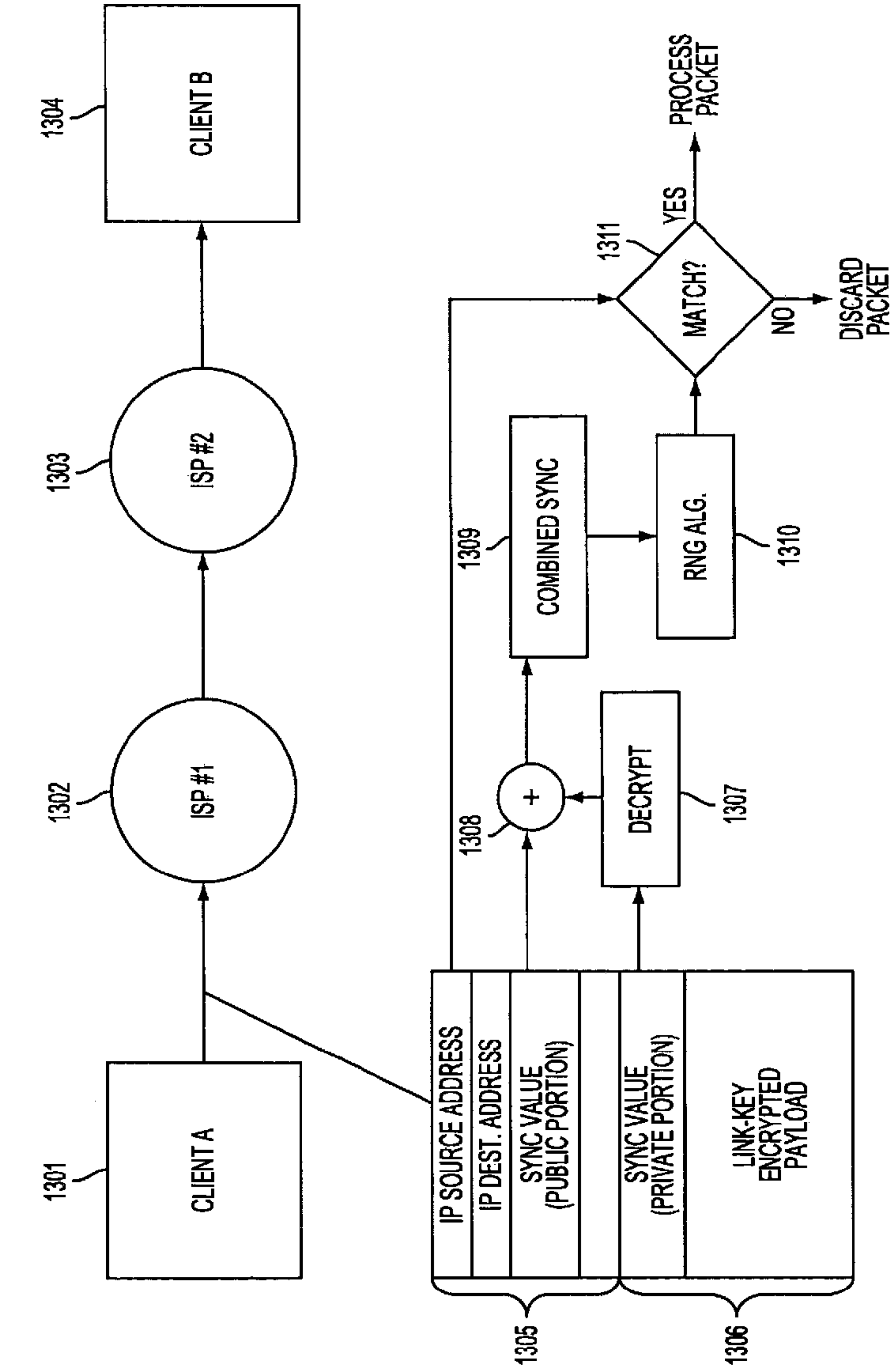

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
1307
DECRYPT
1308
+
1309
COMBINED SYNC
RNG ALG.
1310
1311
MATCH?
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

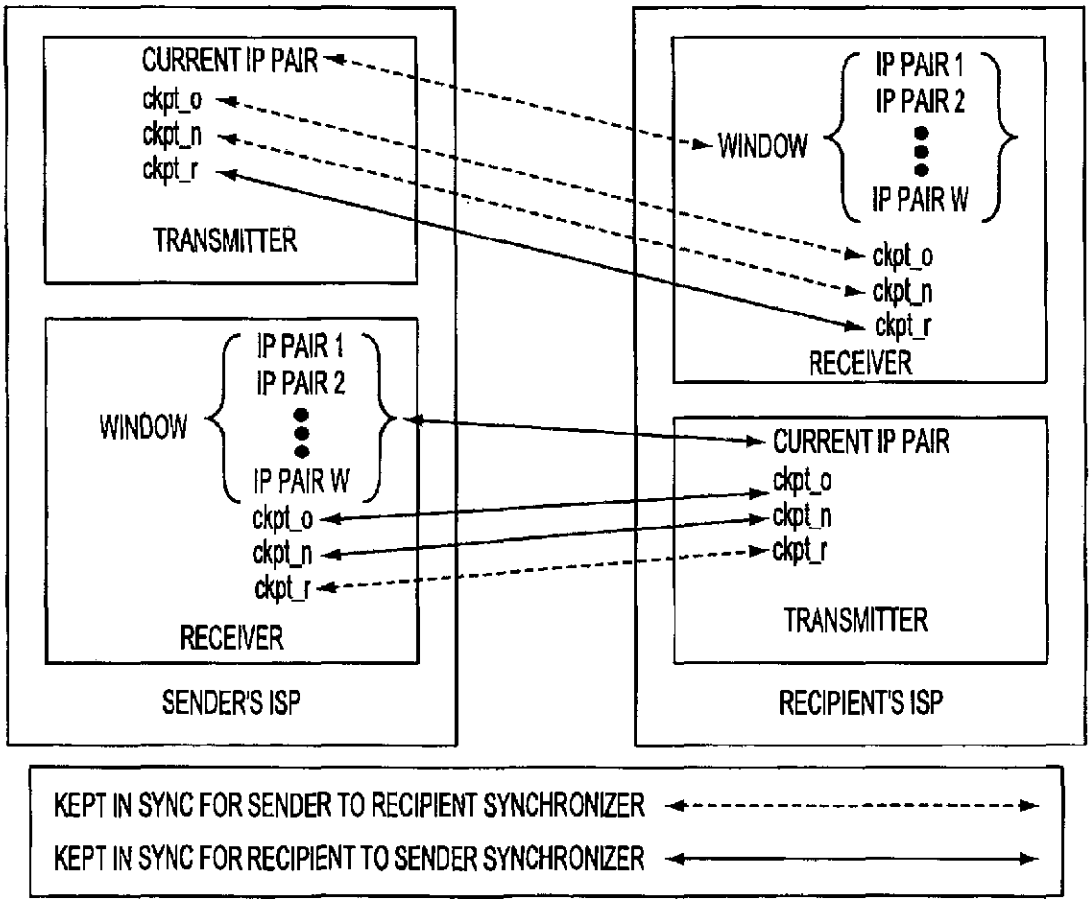

CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
SENDER'S ISP
RECIPIENT'S ISP
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

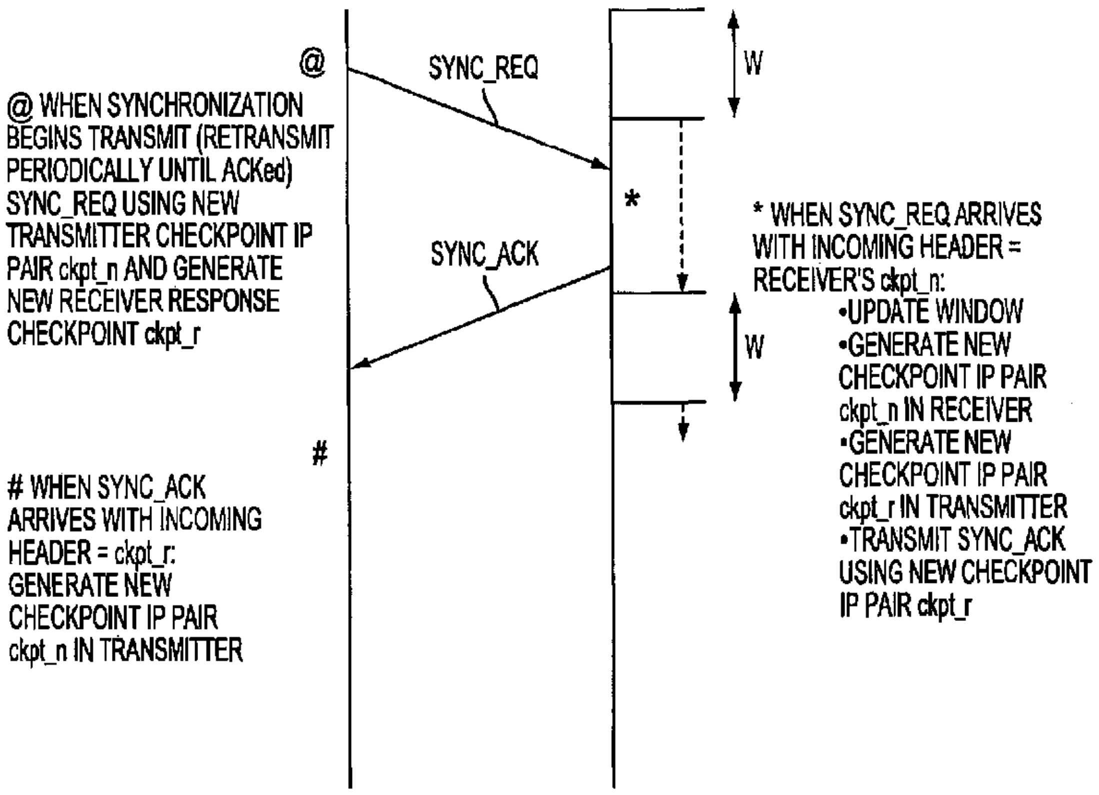

@
SYNC_REQ
W
@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
*
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
•UPDATE WINDOW
•GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
•GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
•TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r
SYNC_ACK
W
#
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

FIG. 15

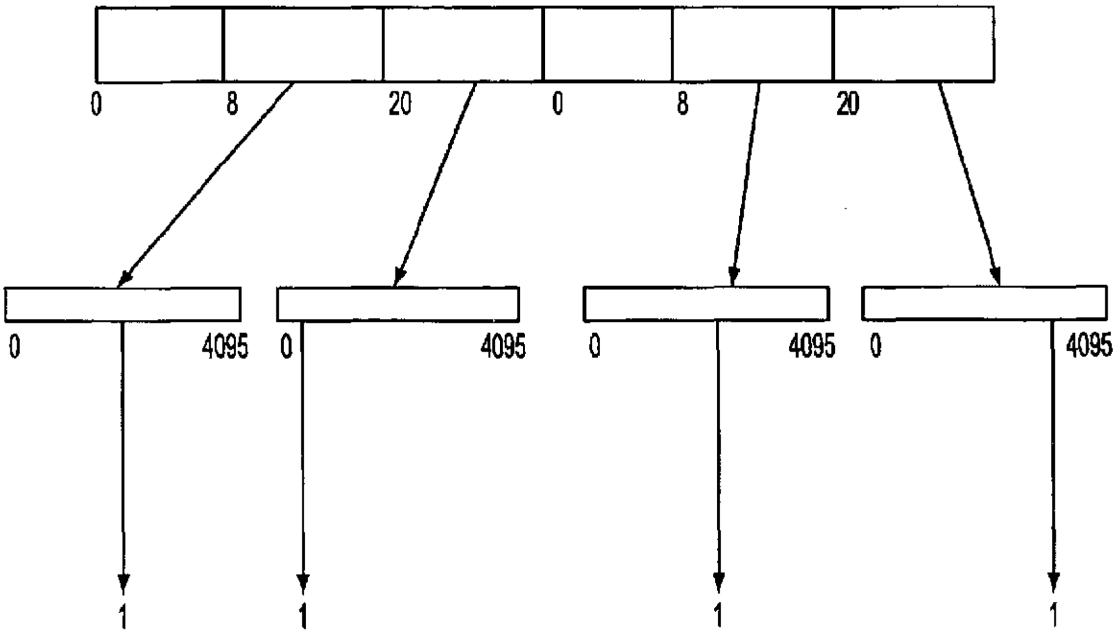


FIG. 16

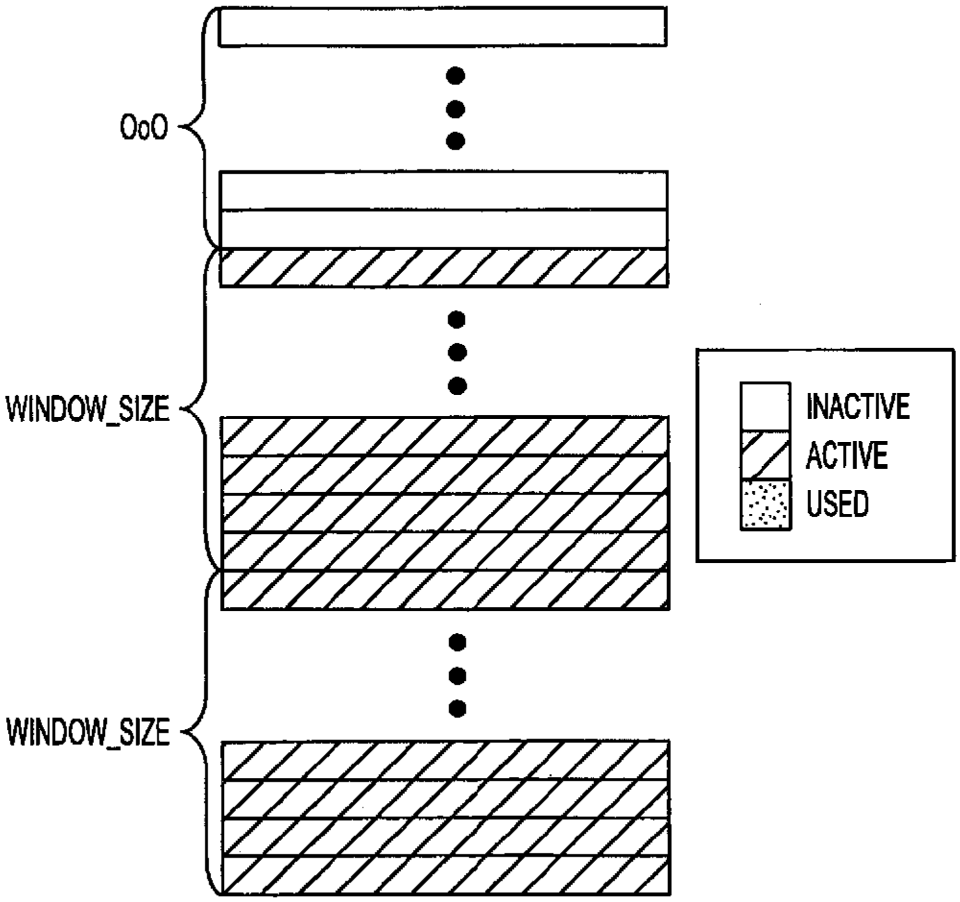


FIG. 17

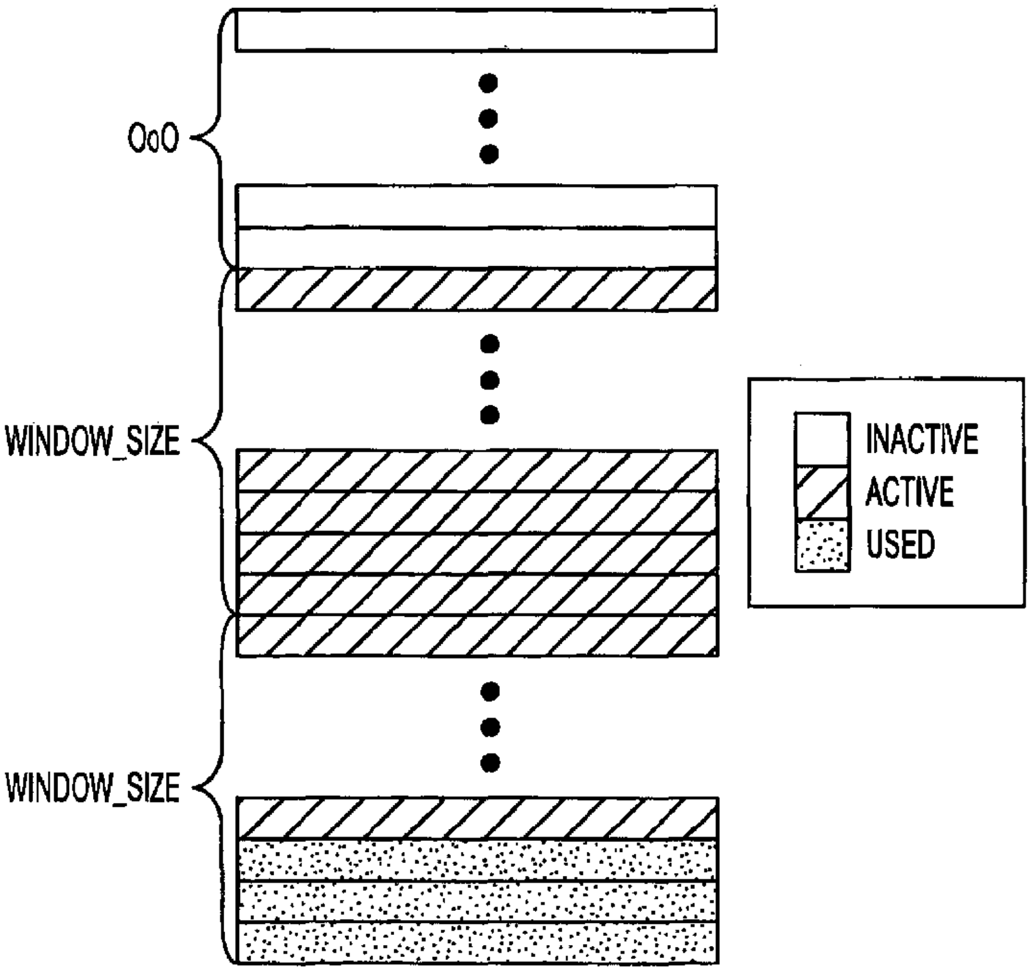


FIG. 18

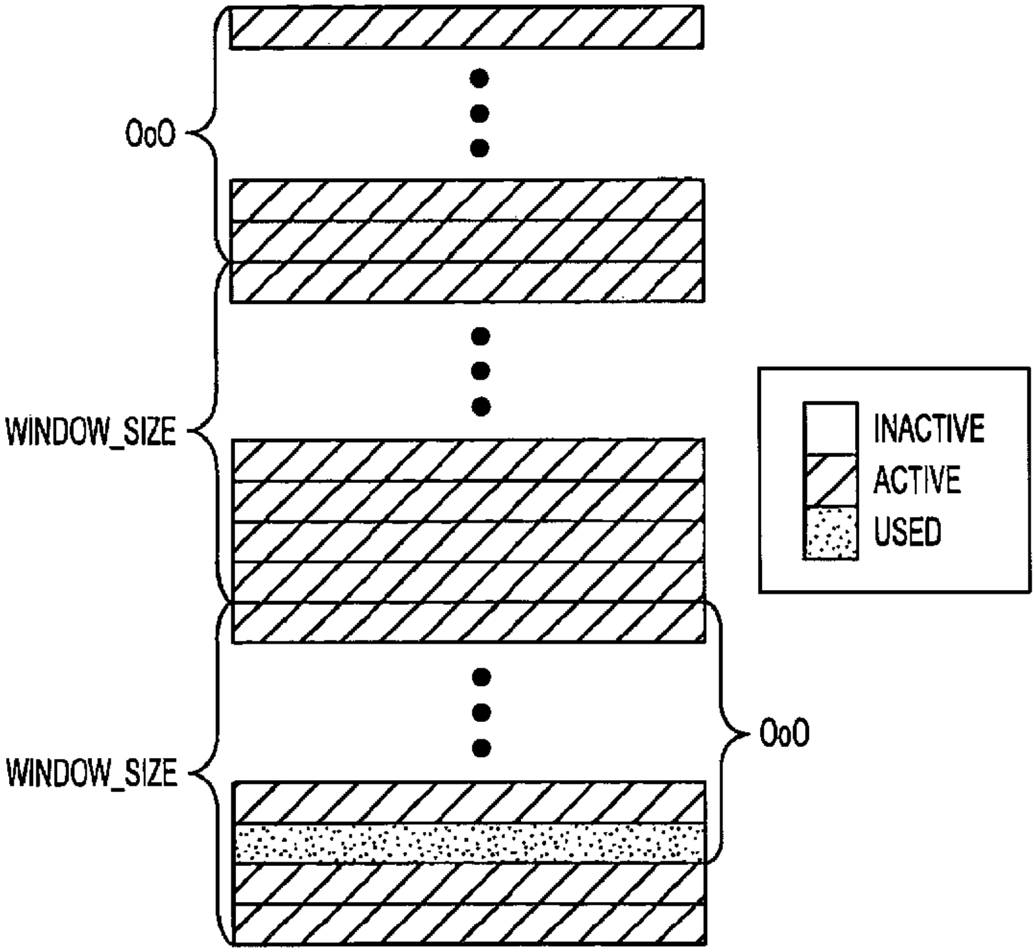


FIG. 19

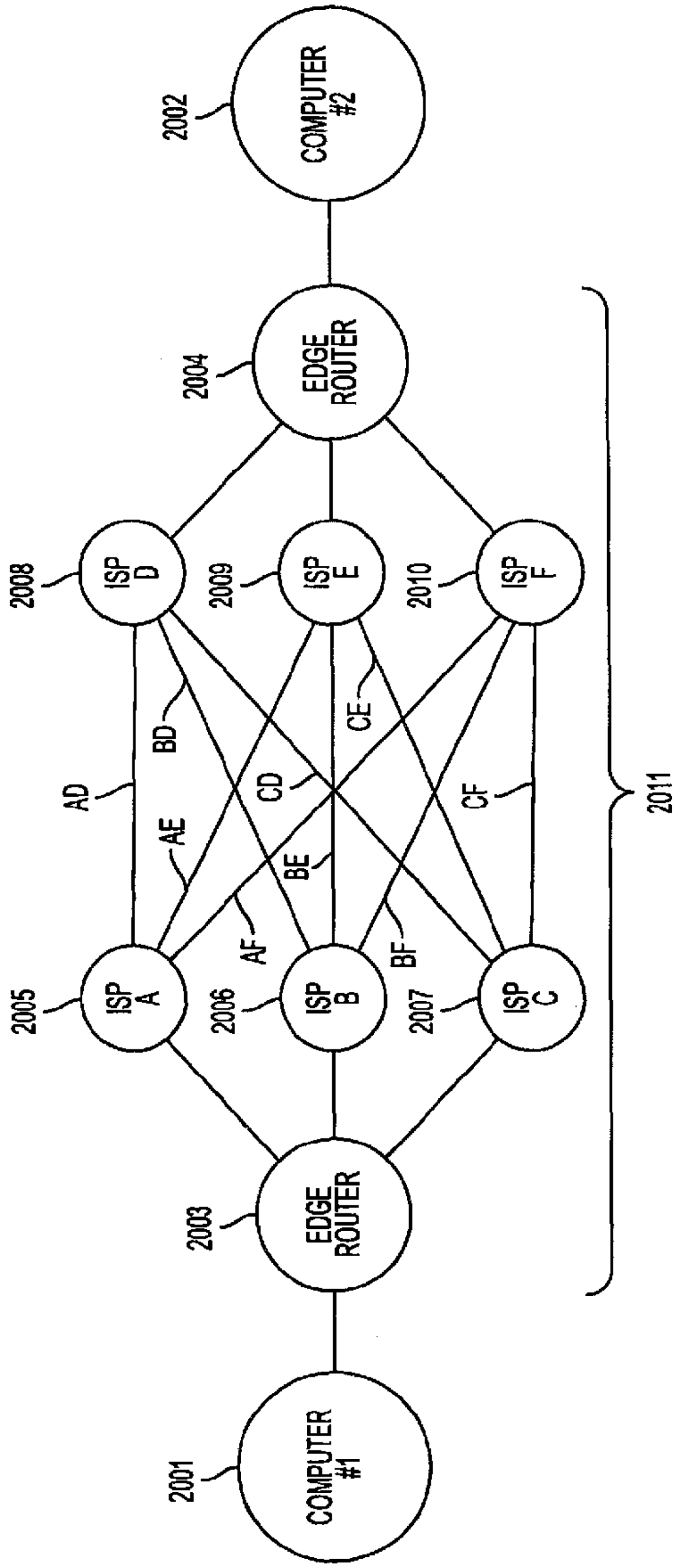

COMPUTER #2
2002
EDGE ROUTER
2004
ISP D
2008
ISP E
2009
ISP F
2010
AD
BD
CD
CE
CF
AE
BE
AF
BF
ISP A
2005
ISP B
2006
ISP C
2007
2011
EDGE ROUTER
2003
COMPUTER #1
2001

FIG. 20

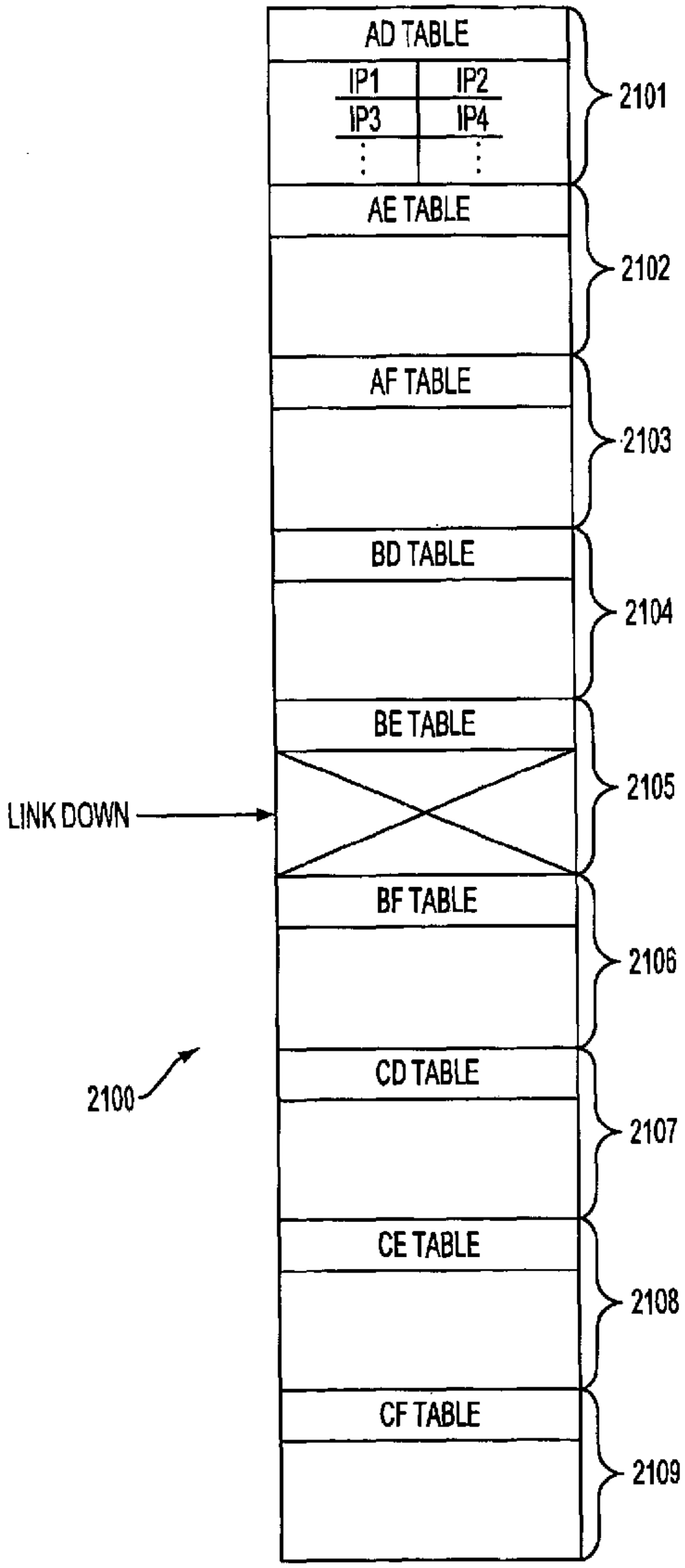

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

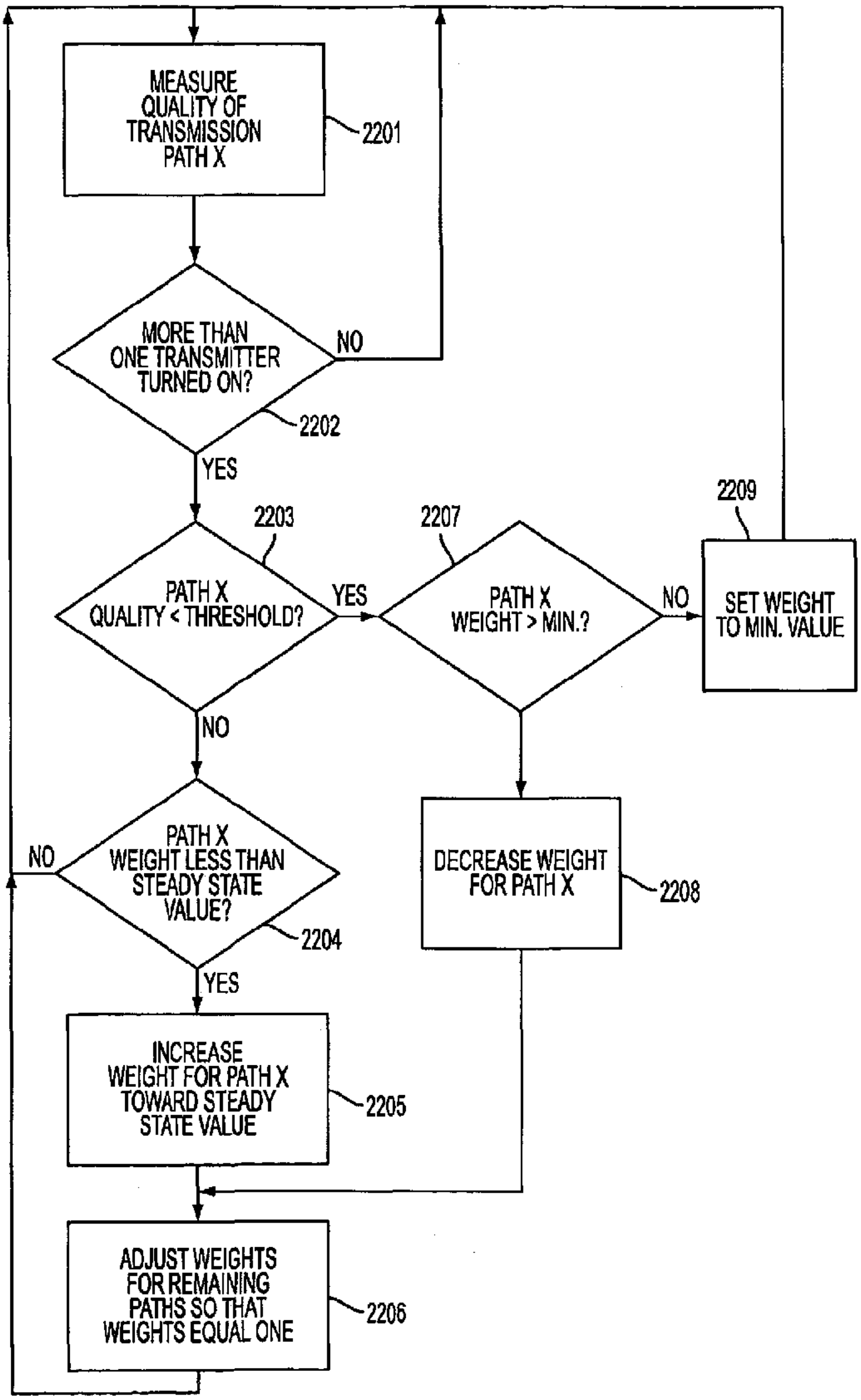

MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
NO
2202
YES
2203
PATH X QUALITY < THRESHOLD?
YES
2207
PATH X WEIGHT > MIN.?
NO
2209
SET WEIGHT TO MIN. VALUE
NO
NO
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
2204
YES
DECREASE WEIGHT FOR PATH X
2208
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

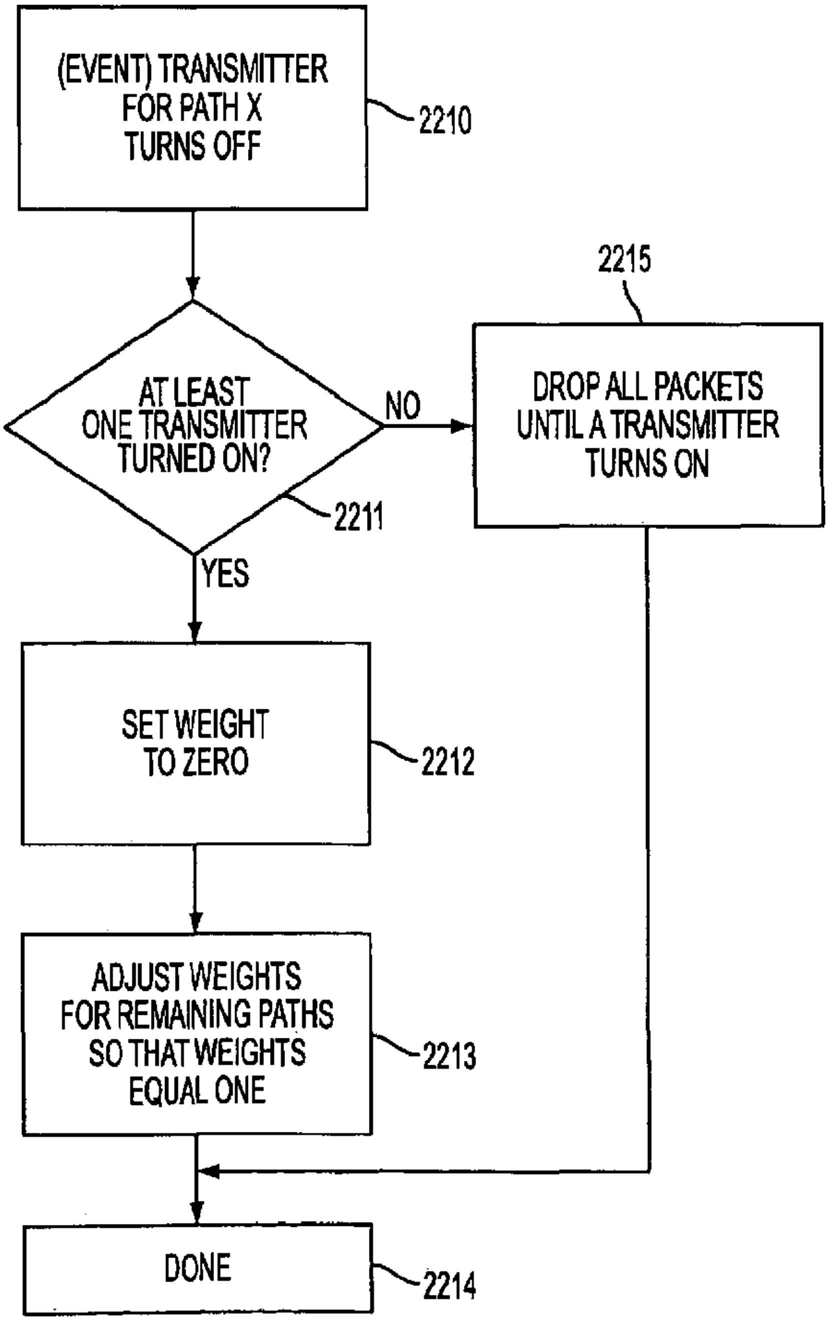

(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2210
AT LEAST ONE TRANSMITTER TURNED ON?
2211
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
SET WEIGHT TO ZERO
2212
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2213
DONE
2214

FIG. 22B

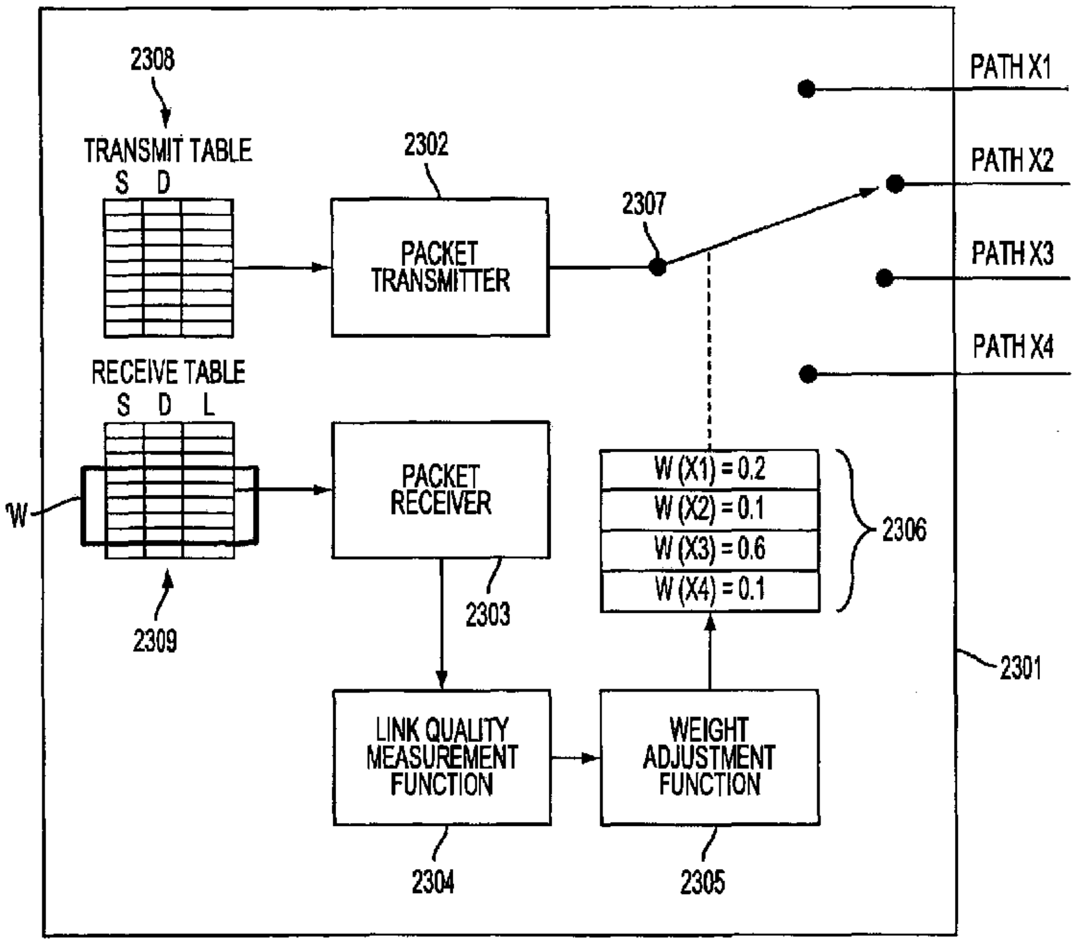

2308
TRANSMIT TABLE
S D
2302
PACKET TRANSMITTER
2307
PATH X1
PATH X2
PATH X3
PATH X4
RECEIVE TABLE
S D L
W
PACKET RECEIVER
2309
2303
W (X1) = 0.2
W (X2) = 0.1
W (X3) = 0.6
W (X4) = 0.1
2306
2301
LINK QUALITY MEASUREMENT FUNCTION
WEIGHT ADJUSTMENT FUNCTION
2304
2305

FIG. 23

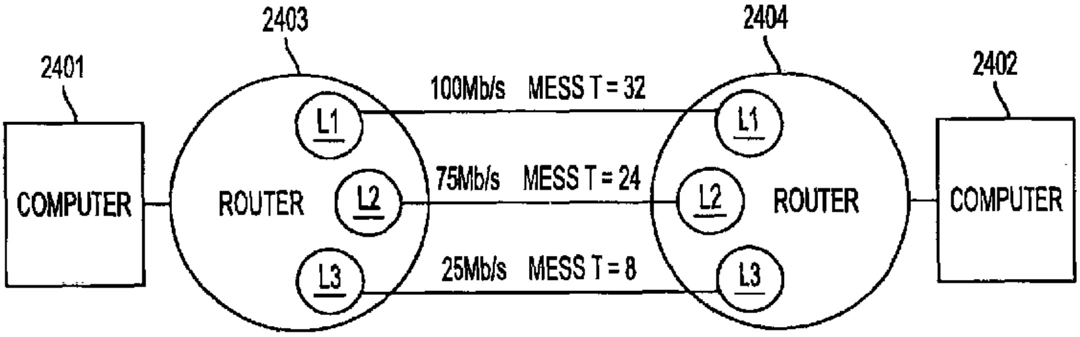

2401
COMPUTER
2403
ROUTER
L1
L2
L3
100Mb/s MESS T = 32
75Mb/s MESS T = 24
25Mb/s MESS T = 8
2404
L1
L2
L3
ROUTER
2402
COMPUTER

FIG. 24

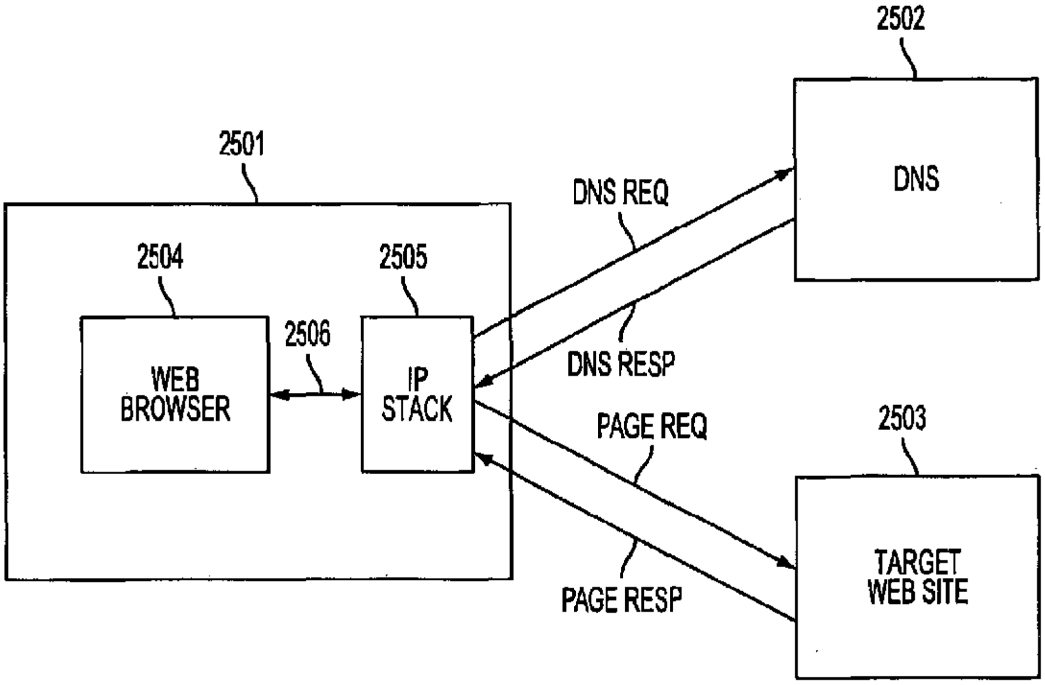


FIG. 25
(PRIOR ART)

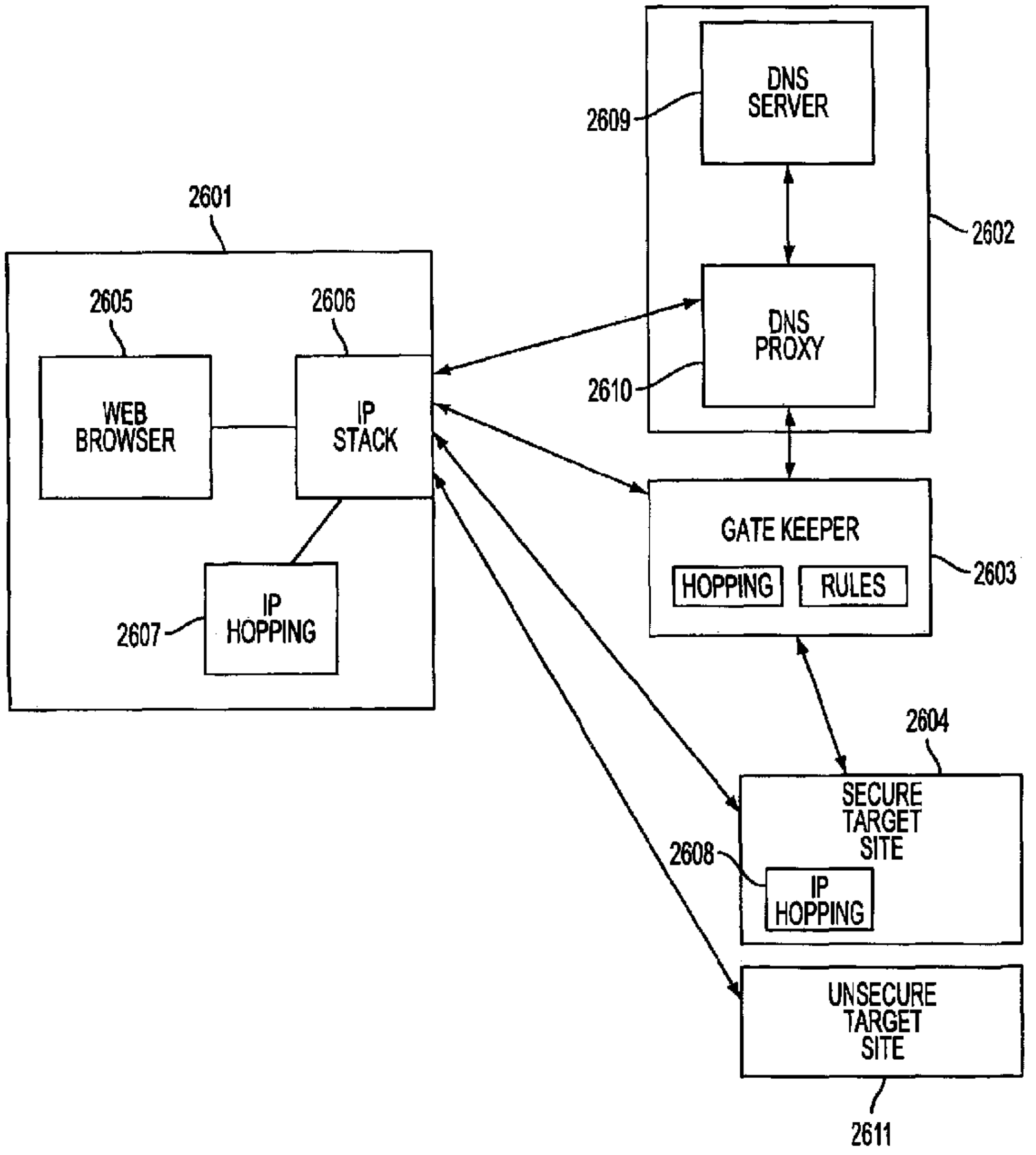

DNS SERVER
2609
2602
DNS PROXY
2610
2601
2605
2606
WEB BROWSER
IP STACK
IP HOPPING
2607
GATE KEEPER
HOPPING
RULES
2603
2604
SECURE TARGET SITE
2608
IP HOPPING
UNSECURE TARGET SITE
2611

FIG. 26

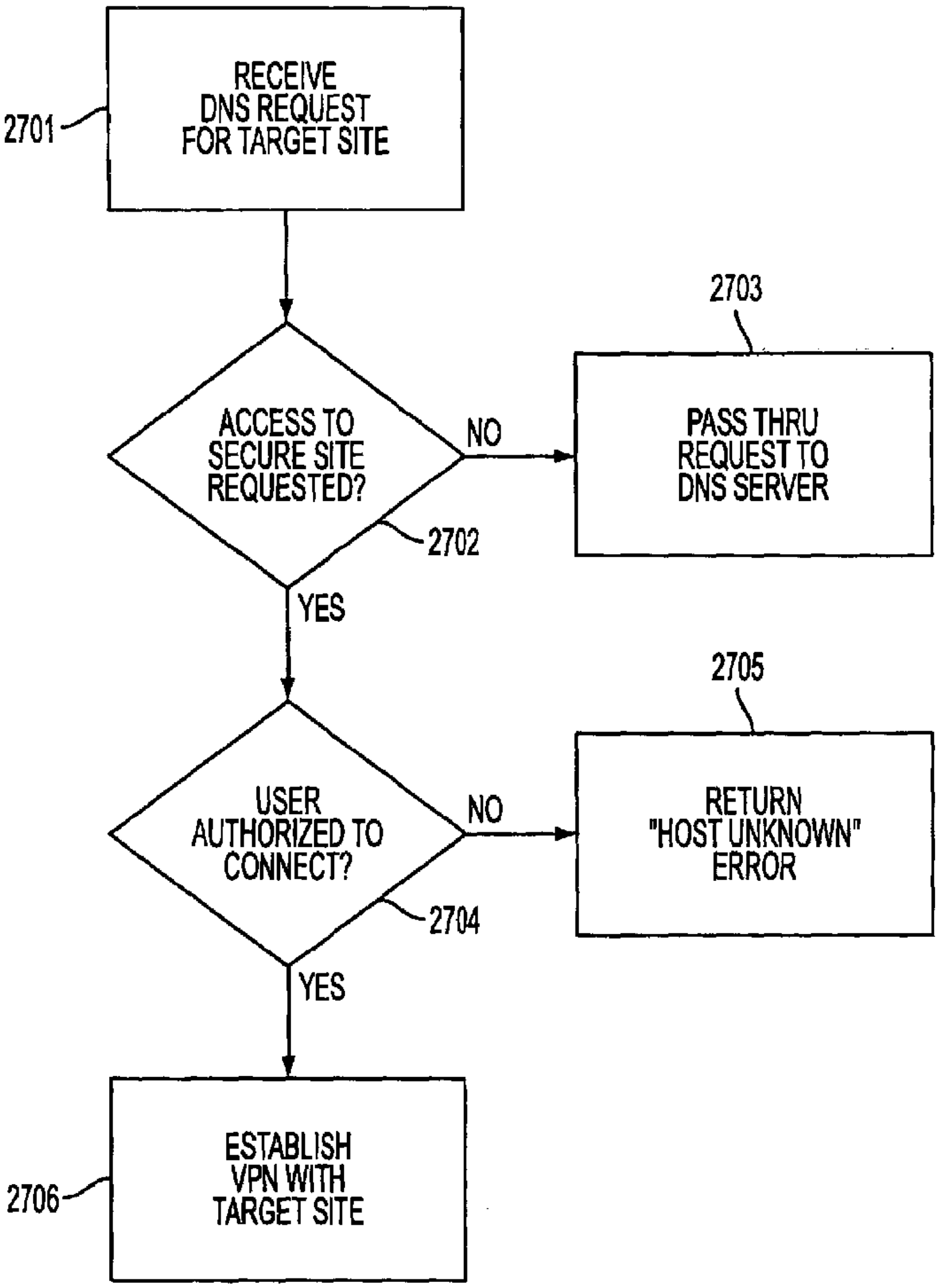

2701
RECEIVE DNS REQUEST FOR TARGET SITE
ACCESS TO SECURE SITE REQUESTED?
2702
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
USER AUTHORIZED TO CONNECT?
2704
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
2706
ESTABLISH VPN WITH TARGET SITE

FIG. 27

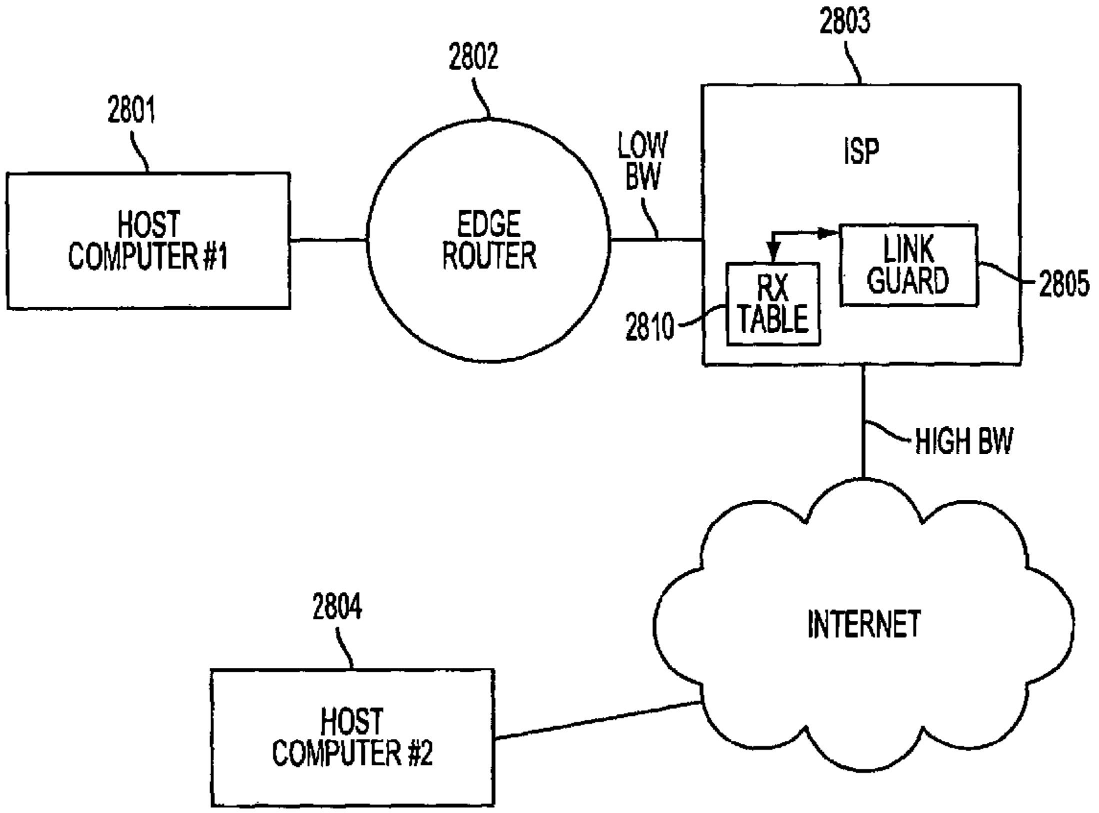

2801
HOST COMPUTER #1
2802
EDGE ROUTER
LOW BW
2803
ISP
LINK GUARD
2805
RX TABLE
2810
HIGH BW
INTERNET
2804
HOST COMPUTER #2

FIG. 28

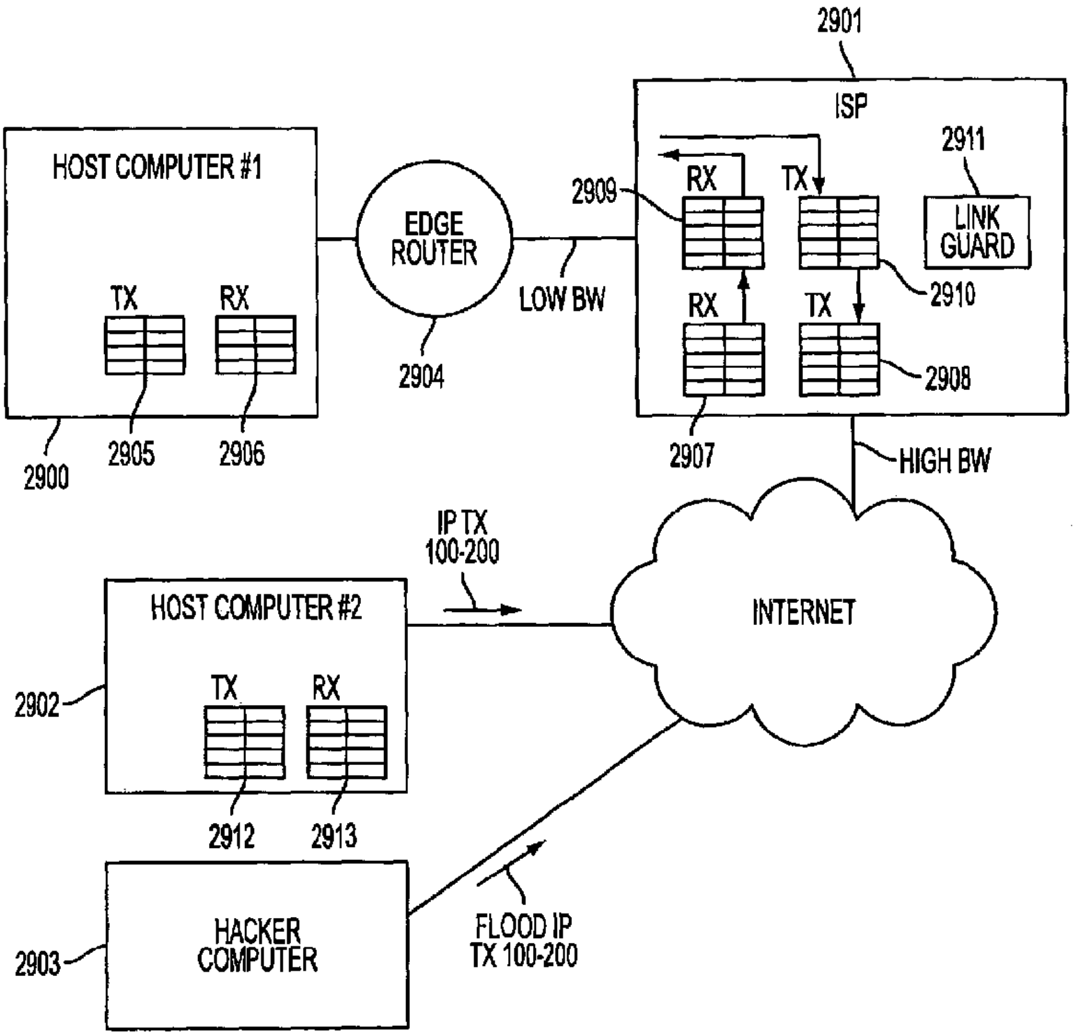

2901
ISP
2911
LINK GUARD
HOST COMPUTER #1
EDGE ROUTER
2909
RX
TX
TX
RX
LOW BW
RX
TX
2910
2904
2908
2900
2905
2906
2907
HIGH BW
IP TX 100-200
HOST COMPUTER #2
INTERNET
2902
TX
RX
2912
2913
HACKER COMPUTER
2903
FLOOD IP TX 100-200

FIG. 29

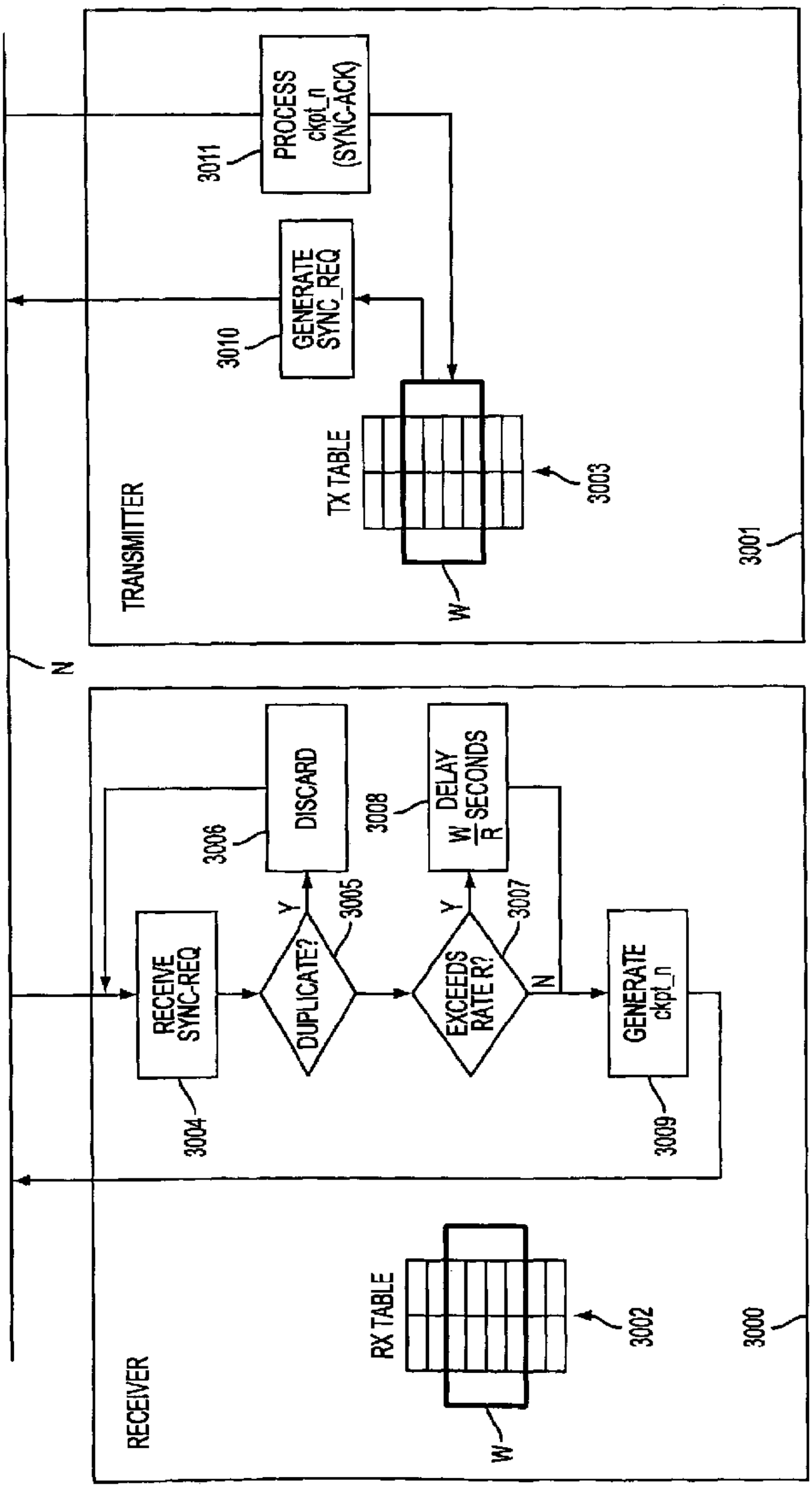

TRANSMITTER
3011
PROCESS ckpt_n (SYNC-ACK)
3010
GENERATE SYNC_REQ
TX TABLE
3003
W
3001
N
RECEIVER
RECEIVE SYNC-REQ
3004
DUPLICATE?
3005
Y
DISCARD
3006
EXCEEDS RATE R?
3007
Y
N
DELAY W/R SECONDS
3008
GENERATE ckpt_n
3009
RX TABLE
3002
W
3000

FIG. 30

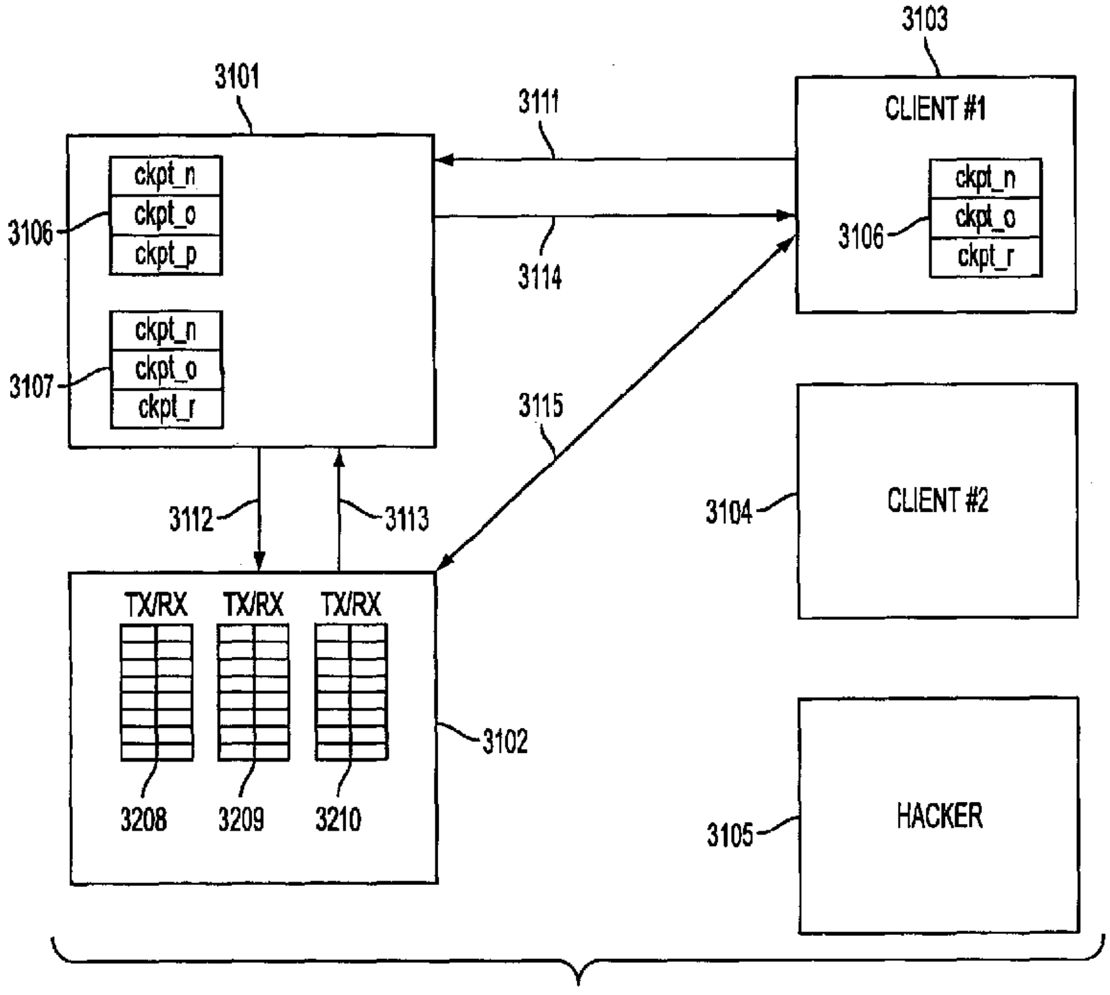

3101
3106
ckpt_n
ckpt_o
ckpt_p
3107
ckpt_n
ckpt_o
ckpt_r
3111
3114
3115
3103
CLIENT #1
3106
ckpt_n
ckpt_o
ckpt_r
3112
3113
TX/RX
TX/RX
TX/RX
3208
3209
3210
3102
3104
CLIENT #2
3105
HACKER

FIG. 31

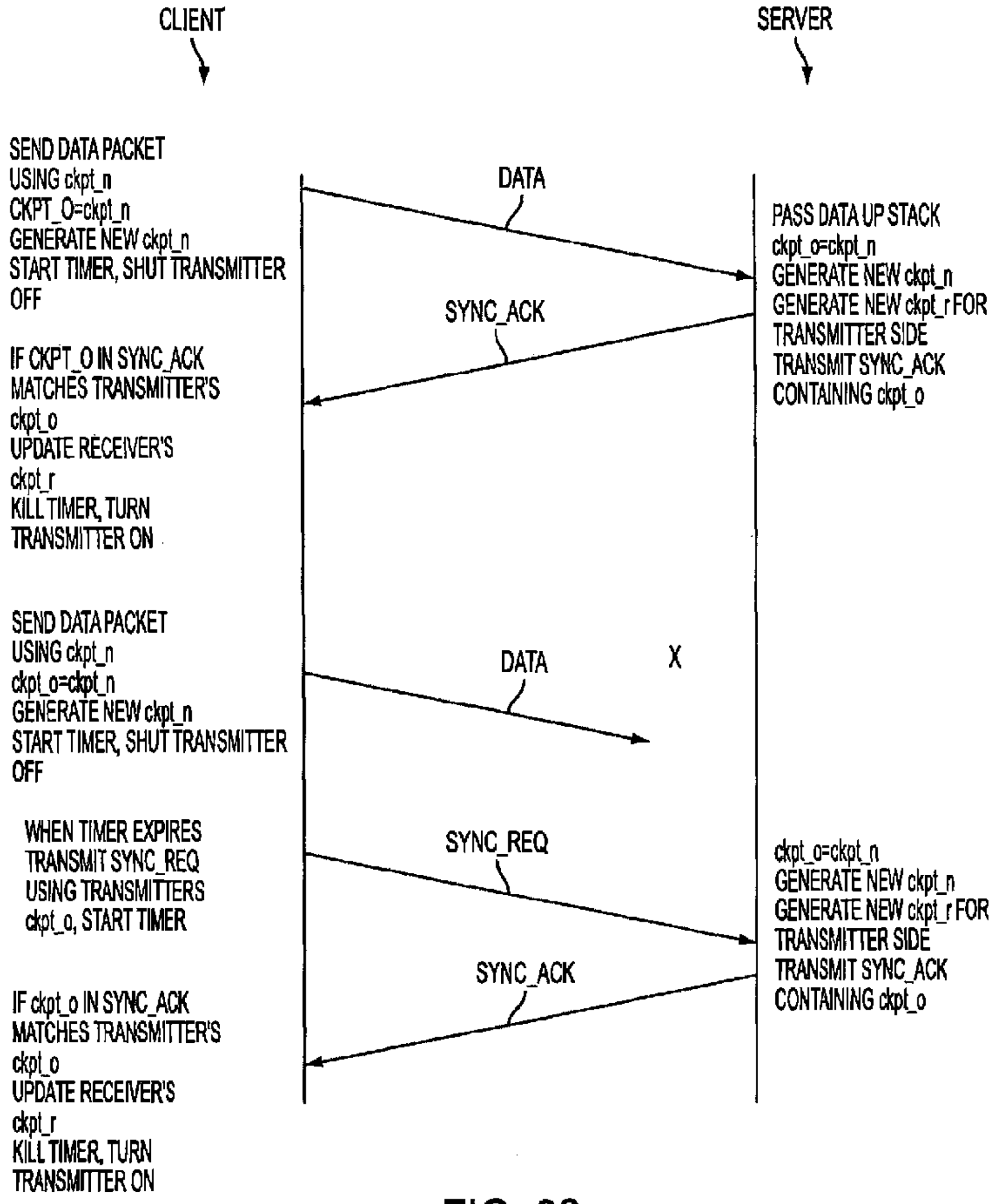

CLIENT
SERVER
SEND DATA PACKET
USING ckpt_n
CKPT_O=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
PASS DATA UP STACK
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING ckpt_n
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
X
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
ckpt_o, START TIMER
SYNC_REQ
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF ckpt_o IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

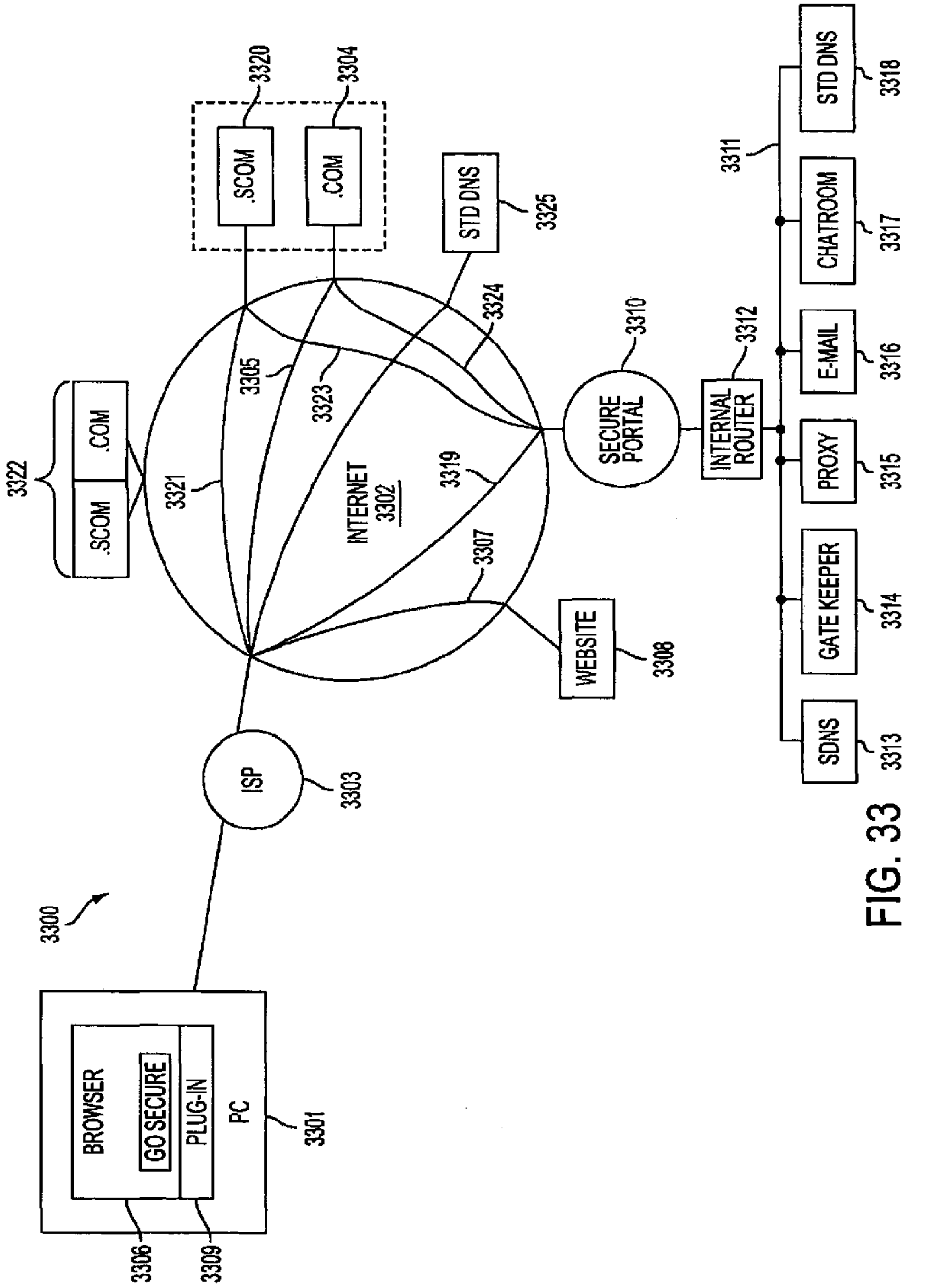

3300
BROWSER
GO SECURE
PLUG-IN
PC
3301
3306
3309
ISP
3303
INTERNET
3302
3321
3305
3323
3319
3307
3324
3322
.SCOM
.COM
3320
3304
STD DNS
3325
WEBSITE
3308
SECURE PORTAL
3310
INTERNAL ROUTER
3312
3311
SDNS
3313
GATE KEEPER
3314
PROXY
3315
E-MAIL
3316
CHATROOM
3317
STD DNS
3318

FIG. 33

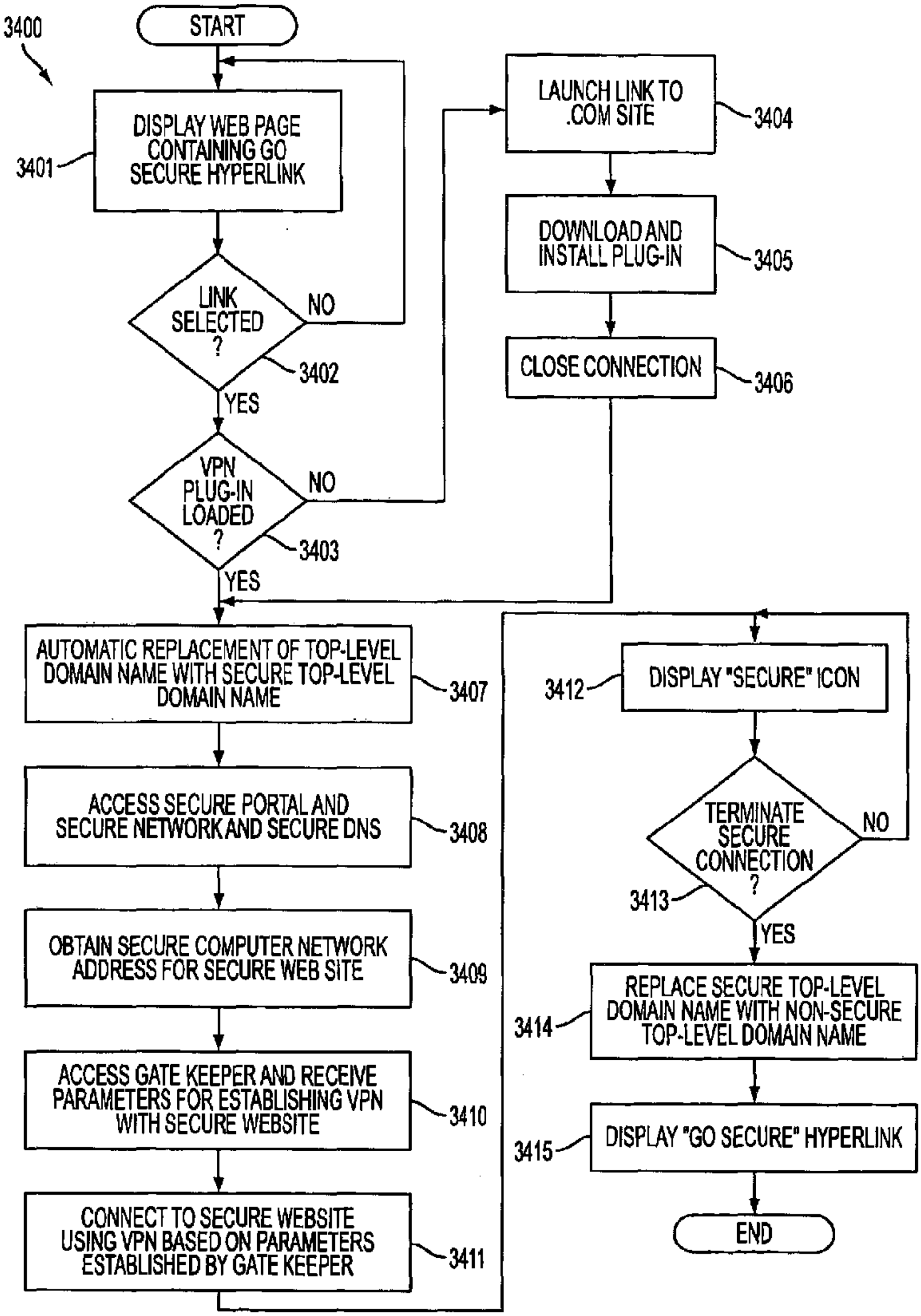

3400
START
DISPLAY WEB PAGE CONTAINING GO SECURE HYPERLINK
3401
LINK SELECTED ?
NO
3402
YES
VPN PLUG-IN LOADED ?
NO
3403
YES
LAUNCH LINK TO .COM SITE
3404
DOWNLOAD AND INSTALL PLUG-IN
3405
CLOSE CONNECTION
3406
AUTOMATIC REPLACEMENT OF TOP-LEVEL DOMAIN NAME WITH SECURE TOP-LEVEL DOMAIN NAME
3407
ACCESS SECURE PORTAL AND SECURE NETWORK AND SECURE DNS
3408
OBTAIN SECURE COMPUTER NETWORK ADDRESS FOR SECURE WEB SITE
3409
ACCESS GATE KEEPER AND RECEIVE PARAMETERS FOR ESTABLISHING VPN WITH SECURE WEBSITE
3410
CONNECT TO SECURE WEBSITE USING VPN BASED ON PARAMETERS ESTABLISHED BY GATE KEEPER
3411
3412
DISPLAY "SECURE" ICON
TERMINATE SECURE CONNECTION ?
NO
3413
YES
REPLACE SECURE TOP-LEVEL DOMAIN NAME WITH NON-SECURE TOP-LEVEL DOMAIN NAME
3414
DISPLAY "GO SECURE" HYPERLINK
3415
END

FIG. 34

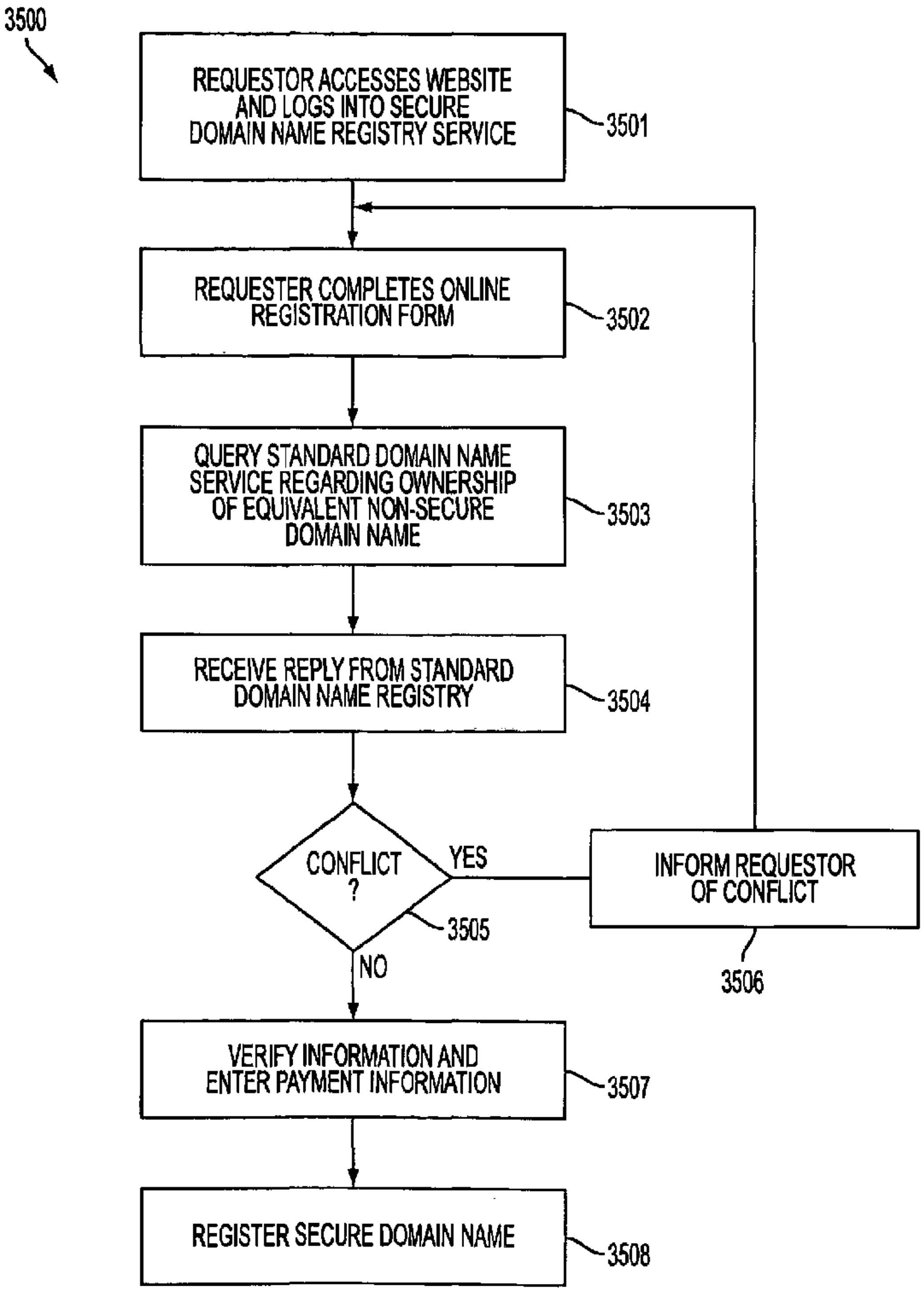

3500
REQUESTOR ACCESSES WEBSITE AND LOGS INTO SECURE DOMAIN NAME REGISTRY SERVICE
3501
REQUESTER COMPLETES ONLINE REGISTRATION FORM
3502
QUERY STANDARD DOMAIN NAME SERVICE REGARDING OWNERSHIP OF EQUIVALENT NON-SECURE DOMAIN NAME
3503
RECEIVE REPLY FROM STANDARD DOMAIN NAME REGISTRY
3504
CONFLICT ?
YES
3505
INFORM REQUESTOR OF CONFLICT
3506
NO
VERIFY INFORMATION AND ENTER PAYMENT INFORMATION
3507
REGISTER SECURE DOMAIN NAME
3508

FIG. 35

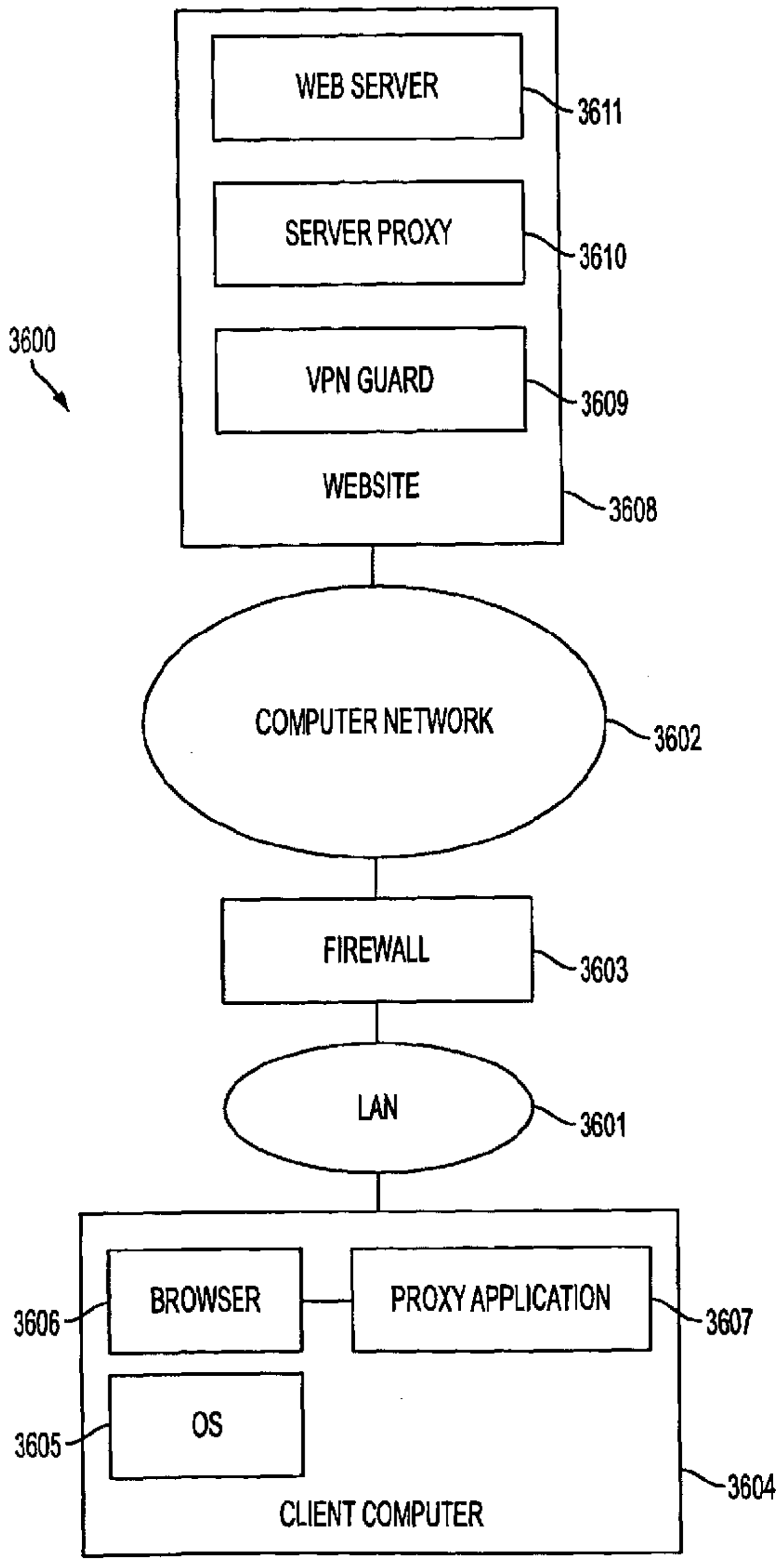

WEB SERVER
3611
SERVER PROXY
3610
3600
VPN GUARD
3609
WEBSITE
3608
COMPUTER NETWORK
3602
FIREWALL
3603
LAN
3601
3606
BROWSER
PROXY APPLICATION
3607
3605
OS
3604
CLIENT COMPUTER

FIG. 36

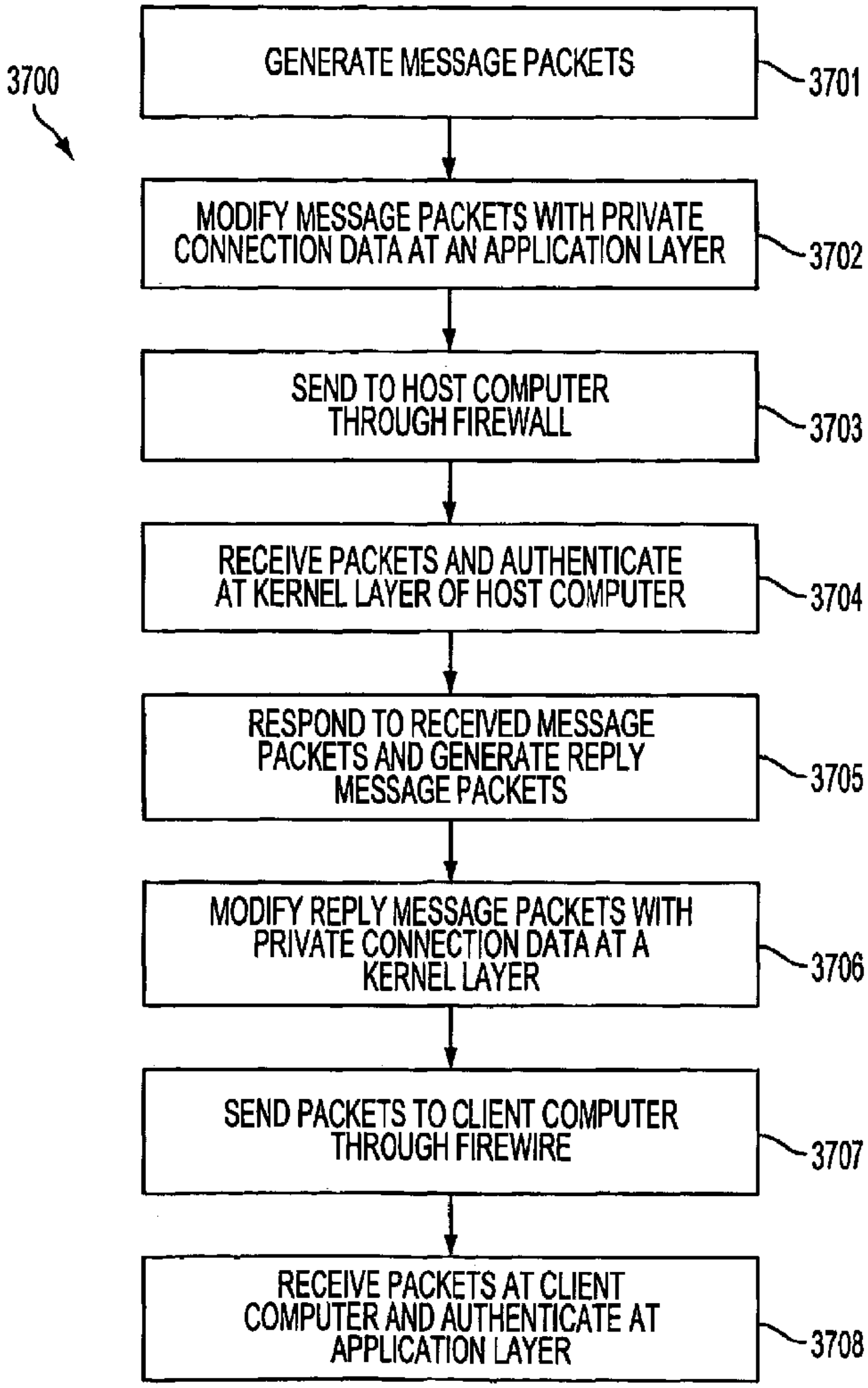

3700
GENERATE MESSAGE PACKETS
3701
MODIFY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT AN APPLICATION LAYER
3702
SEND TO HOST COMPUTER THROUGH FIREWALL
3703
RECEIVE PACKETS AND AUTHENTICATE AT KERNEL LAYER OF HOST COMPUTER
3704
RESPOND TO RECEIVED MESSAGE PACKETS AND GENERATE REPLY MESSAGE PACKETS
3705
MODIFY REPLY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT A KERNEL LAYER
3706
SEND PACKETS TO CLIENT COMPUTER THROUGH FIREWIRE
3707
RECEIVE PACKETS AT CLIENT COMPUTER AND AUTHENTICATE AT APPLICATION LAYER
3708

FIG. 37

AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from and is a continuation patent application of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000 now abandoneed, which is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999 now U.S. Pat. No. 7,010,604. The subject matter of U.S. application Ser. No. 09/429,643, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, and which is incorporated by reference herein.

GOVERNMENT CONTRACT RIGHTS

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and 110, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or

"reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet transmitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply information packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet. Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12**A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12**B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. **20** shows a system employing distributed transmission paths.

FIG. **21** shows a plurality of link transmission tables that can be used to route packets in the system of FIG. **20**.

FIG. **22**A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. **22**B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a couventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

FIG. 33 shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. 34 shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. 35 shows a flow diagram for registering a secure domain name according to the present invention.

FIG. 36 shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. 37 shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryptiou in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP_C. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUT's. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IP_T are formed. The TARP headers IP_T can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IP_T are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IP_T, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP_C is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. 4, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122**-**127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IPT headers are converted into normal IPc headers. The window sequence numbers are integrated in the IPC headers.

S50. The packets are then handed up to the IP layer processes.

1. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily he different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. **9**; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. **10**, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. **11**, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP1 and IP2, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP3. Each frame header generally includes a source hardware address **1101A** and a destination hardware address **10B**; other well-known fields in frame headers are omitted from FIG. **11** for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. **12**A and **12**B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208**X that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222**X is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window W3 that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window W3 slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window W3 will be accepted; those falling outside of window W3 will be rejected as invalid. The length of window W3 can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221**X, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209**X. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window W1 maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12**A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210**X to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101**A and **1101**B in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12**B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead-since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be

prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair; this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which

causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers X_1, X_2, X_3 . . . X_k starting with seed X_0 using a recurrence

$$X_i=(a\,X_{i-1}+b) \bmod c, \tag{1}$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1)) \bmod c \tag{2}$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i-(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c. \tag{3}$$

It can be shown that:

$$(a_i(X_0(a-1)+b)-b)/(a-1) \bmod c=((a^i \bmod ((a-1)c)(X_0(a-1)+b)-b)/(a-1)) \bmod c \tag{4}.$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b)$ mod c, b as b mod c and compute a^i mod((a−1)c) (this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the j^{th} checkpoint, as X_0 and n as i, a node can store a^nmod((a−1)c) once per LCR and set

$$X_{j+1}^w=X_{n(j+1)}=((a^n \bmod ((a-1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \tag{5}$$

to generate the random number for the $j+1^{th}$ synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4) \bmod 15. \tag{6}$$

If one sets X_0=1, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, c*(a−1)=15*30=450 and a^n mod ((a−1)c)=31^3mod(15*30)=29791mod(450)=91. Equation (5) becomes:

$$((91\,(X_i30+4)-4)/30) \bmod 15 \tag{7}.$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91\ (X_i30 + 4) - 4$	$((91\ (X_i30 + 4) - 4)/30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not he received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^{n-1}). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will he rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses (1≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧2). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. 17 depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. 18 when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. 19 shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE–OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. 20. In this embodiment, a message transmission system includes a first computer 2001 in communication with a second computer 2002 through a network 2011 of intermediary computers. In one variant of this embodiment, the network includes two edge routers 2003 and 2004 each of which is linked to a plurality of Internet Service Providers (ISPs) 2005 through 2010. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. 20, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. 20 to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element 2005) to ISP D (element 2008)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. 21, computer 2001 or edge router 2003 incorporates a plurality of link transmission tables 2100 that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table 2101 contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer 2001 to second computer 2002, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element 2005) and ISP B (element 2008)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table 2105, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. Continuation-In-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. 20 and 21 and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step 2201, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step 2202, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step 2201.

In step 2203, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step 2207 a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step 2209 the weight is set to the minimum level and processing resumes at step 2201. If the weight is above the minimum level, then in step 2208 the weight is gradually decreased for the path, then in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step 2203 the quality of the path was greater than or equal to the threshold, then in step 2204 a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step 2205 the weight is increased toward the steady-state value, and in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step 2204 the weight is not less than the steady-state value, then processing resumes at step 2201 without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. 22A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step 2210, a transmitter shut-down event occurs. In step 2211, a test is made to determine whether at least one transmitter is still turned on. If not, then in step 2215 all packets are dropped until a transmitter turns on. If in step 2211 at least one transmitter is turned on, then in step 2212 the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. 23 shows a computer node 2301 employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. 23 communicate over a plurality of separate physical transmission paths. As shown in FIG. 23, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter 2302 that operates in accordance with a transmit table 2308 as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver 2303 that operates in accordance with a receive table 2309, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table 2308 according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch 2307. Switch 2307, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table 2306. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to asses the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P'=\alpha\times MIN+(1-\alpha)\times P \qquad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P'=\beta\times S+(1-\beta)\times P \qquad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.

2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.

3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.

4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.

5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.

6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. 25. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **2704**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **2703** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **2701**, a DNS look-up request is received for a target host. In step **2702**, a check is made to determine whether access to a secure host was requested. If not, then in step **2703** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **2702**, if access to a secure host was requested, then in step **2704** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **2705**). If the user has sufficient security privileges, then in step **2706** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permissiou to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses **100** to **200** for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will reissue the SYNC_REQ every T1 seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of M×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×W/R seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from thetransmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million

subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer TI noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.

2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and and other information (i.e., user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.

4. TI expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPT_O associated with the timer, it starts timer Ti noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.

5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

F. One-click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. 33 shows a system block diagram 3300 of a computer network in which the one-click secure communication method of the present invention is suitable. In FIG. 33, a computer terminal or client computer 3301, such as a personal computer (PC), is connected to a computer network 3302, such as the Internet, through an ISP 3303. Alternatively, computer 3301 can be connected to computer network 3302 through an edge router. Computer 3301 includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer 3301 can communicate conventionally with another computer 3304 connected to computer network 3302 over a communication link 3305 using a browser 3306 that is installed and operates on computer 3301 in a well-known manner.

Computer 3304 can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network 3302 is the Internet, computer 3304 typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. 34 shows a flow diagram 3400 for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step 3401, computer 3301 is connected to server computer 3304 over a non-VPN communication link 3305. Web browser 3306 displays a web page associated with server 3304 in a well-known manner. According to one variation of the invention, the display of computer 3301 contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network 3302 between terminal 3301 and server 3304. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer 3304, thereby indicating that the entity providing server 3304 also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer 3301 is informed that the current communication link between computer 3301 and server computer 3304 is a non-secure, non-VPN communication link. At step 3402, it is determined whether a user of computer 3301 has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step 3402, it is determined that the user has selected the "go secure" hyperlink, flow continues to step 3403 where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer 3301. Alternatively, a user can enter a command into computer 3301 to "go secure."

If, at step 3403, the object determines that the software module has been installed, flow continues to step 3407. If, at step 3403, the object determines that the software module has not been installed, flow continues to step 3404 where a non-VPN communication link 3307 is launched between computer 3301 and a website 3308 over computer network 3302 in a well-known manner. Website 3308 is accessible by all computer terminals connected to computer network 3302 through a non-VPN communication link. Once connected to

website 3308, a software module for establishing a secure communication link over computer network 3302 can be downloaded and installed. Flow continues to step 3405 where, after computer 3301 connects to website 3308, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal 3301 as software module 3309. At step 3405, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network 3302. At step 3406, the communication link between computer 3301 and website 3308 is then terminated in a well-known manner.

By clicking on the "go secure" hyperlink, a user at computer 3301 has enabled a secure communication mode of communication between computer 3301 and server computer 3304. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer 3301 and server computer 3304 are performed transparently to a user at computer 3301.

At step 3407, a secure VPN communications mode of operation has been enabled and software module 3309 begins to establish a VPN communication link. In one embodiment, software module 3309 automatically replaces the top-level domain name for server 3304 within browser 3406 with a secure top-level domain name for server computer 3304. For example, if the top-level domain name for server 3304 is .com, software module 3309 replaces the .com top-level domain name with a scom top-level domain name, where the "s" stands for secure. Alternatively, software module 3409 can replace the top-level domain name of server 3304 with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module 3309 contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module 3309 accesses a secure portal 3310 that interfaces a secure network 3311 to computer network 3302. Secure network 3311 includes an internal router 3312, a secure domain name service (SDNS) 3313, a VPN gatekeeper 3314 and a secure proxy 3315. The secure network can include other network services, such as e-mail 3316, a plurality of chatrooms (of which only one chatroom 3317 is shown), and a standard domain name service (STD DNS) 3318. Of course, secure network 3311 can include other resources and services that are not shown in FIG. 33.

When software module 3309 replaces the standard top-level domain name for server 3304 with the secure top-level domain name, software module 3309 sends a query to SDNS 3313 at step 3408 through secure portal 3310 preferably using an administrative VPN communication link 3319. In this configuration, secure portal 3310 can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal 3310 authenticates the query from software module 3309 based on the particular information hopping technique used for VPN communication link 3319.

SDNS 3313 contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS 3313 stores a computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS 3313 so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS 3313 for the secure computer network address for the securities trading website, SDNS 3313 determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step 3409, SDNS 3313 accesses VPN gatekeeper 3314 for establishing a VPN communication link between software module 3309 and secure server 3320. Server 3320 can only be accessed through a VPN communication link. VPN gatekeeper 3314 provisions computer 3301 and secure web server computer 3320, or a secure edge router for server computer 3320, thereby creating the VPN. Secure server computer 3320 can be a separate server computer from server computer 3304, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer 3322. Returning to FIG. 34, in step 3410, SDNS 3313 returns a secure URL to software module 3309 for the .scom server address for a secure server 3320 corresponding to server 3304.

Alternatively, SDNS 3313 can be accessed through secure portal 3310 "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal 3310 preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS 3319. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS 3313 can be in the clear, and SDNS 3313 and gatekeeper 3314 can operate to establish a VPN communication link to the querying computer for sending the reply.

At step 3411, software module 3309 accesses secure server 3320 through VPN communication link 3321 based on the VPN resources allocated by VPN gatekeeper 3314. At step 3412, web browser 3306 displays a secure icon indicating that the current communication link to server 3320 is a secure VPN communication link. Further communication between computers 3301 and 3320 occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link 3321 is terminated at step 3413, flow continues to step 3414 where software module 3309 automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server 3304. Browser 3306 accesses a standard DNS 3325 for obtaining the non-secure URL for server 3304. Browser 3306 then connects to server 3304 in a well-known manner. At step 3415, browser 3306 displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal 3301 and server 3304. By again displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module 3309 is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network 3302 are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module 3309 transparently accesses SDNS 3313 for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to be effected.

Additionally, a user at computer 3301 can optionally select a secure communication link through proxy computer 3315. Accordingly, computer 3301 can establish a VPN communication link 3323 with secure server computer 3320 through proxy computer 3315. Alternatively, computer 3301 can establish a non-VPN communication link 3324 to a non-secure website, such as non-secure server computer 3304.

FIG. 35 shows a flow diagram 3500 for registering a secure domain name according to the present invention. At step 3501, a requester accesses website 3308 and logs into a secure domain name registry service that is available through website 3308. At step 3502, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step 3503, the secure domain name registry service at website 3308 queries a non-secure domain name server database, such as standard DNS 3322, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step 3504, the secure domain name registry service at website 3308 receives a reply from standard DNS 3322 and at step 3505 determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step 3507, otherwise flow continues to step 3506 where the requestor is informed of the conflicting ownership information. Flow returns to step 3502.

When there is no conflicting ownership information at step 3505, the secure domain name registry service (website 3308) informs the requestor that there is no conflicting ownership information and prompts the requester to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step 3508 where the newly registered secure domain name sent to SDNS 3313 over communication link 3326.

If, at step 3505, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requester for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS 3325 in a well-known manner. Flow then continues to step 3508.

G. Tunneling Secure Address Hopping Protocol Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. 36 shows a system block diagram of a computer network 3600 in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. 37 shows a flow diagram 3700 for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. 36 a local area network (LAN) 3601 is connected to another computer network 3602, such as the Internet, through a firewall arrangement 3603. Firewall arrangement operates in a well-known manner to interface LAN 3601 to computer network 3602 and to protect LAN 3601 from attacks initiated outside of LAN 3601.

A client computer 3604 is connected to LAN 3601 in a well-known manner. Client computer 3604 includes an operating system 3605 and a web browser 3606. Operating system 3605 provides kernel mode functions for operating client computer 3604. Browser 3606 is an application program for accessing computer network resources connected to LAN 3601 and computer network 3602 in a well-known manner. According to the present invention, a proxy application 3607 is also stored on client computer 3604 and operates at an application layer in conjunction with browser 3606. Proxy application 3607 operates at the application layer within client computer 3604 and when enabled, modifies unprotected, unencrypted message packets generated by browser 3606 by inserting data into the message packets that are used for forming a virtual private connection between client computer 3604 and a server computer connected to LAN 3601 or computer network 3602. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application 3607 is conveniently installed and uninstalled hy a user because proxy application 3607 operates at the application layer within client computer 3604. On installation, proxy application 3607 preferably configures browser 3606 to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer 3604 and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application 3606 can be selected and enabled through, for example, an option provided by browser 3606. Additionally, proxy application 3607 can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer 3604 and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. 37, at step 3701, unprotected and unencrypted message packets are generated by browser 3606. At step 3702, proxy application 3607 modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer 3604 and a destination server computer into the payload portion. At step, 3703, the modified message packets are sent from client computer 3604 to, for example, website (server computer) 3608 over computer network 3602.

Website 3608 includes a VPN guard portion 3609, a server proxy portion 3610 and a web server portion 3611. VPN guard portion 3609 is embedded within the kernel layer of the operating system of website 3608 so that large bandwidth attacks on website 3608 are rapidly rejected. When client computer 3604 initiates an authenticated connection to website 3608, VPN guard portion 3609 is keyed with the hopping sequence contained in the message packets from client computer 3604, thereby performing a strong authentication of the client packet streams entering website 3608 at step 3704. VPN guard portion 3609 can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion 3609 can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion 3609 would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

Server proxy portion **3610** also operates at the kernel layer within website **3608** and catches incoming message packets from client computer **3604** at the VPN level. At step **3705**, server proxy portion **3610** authenticates the message packets at the kernel level within host computer **3604** using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion **3611** as normal TCP web transactions.

At step **3705**, web server portion **3611** responds to message packets received from client computer **3604** in accordance with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion **3611** generates message packets corresponding to the requested webpage. At step **3706**, the reply message packets pass through server proxy portion **3610**, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer **3608** and client computer **3604** over computer network **3602**. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion **3610** operates at the kernel layer within host computer **3608** to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer **3608** to client computer **3604** conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step **3707**, the modified packets are sent from host computer **3608** over computer network **3602** and pass through firewall **3603**. Once through firewall **3603**, the modified packets are directed to client computer **3604** over LAN **3601** and are received at step **3708** by proxy application **3607** at the application layer within client computer **3604**. Proxy application **3607** operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising:

a domain name service system configured to be connected to a communication network, to store a plurality of domain names and corresponding network addresses, to receive a query for a network address, and to comprise an indication that the domain name service system supports establishing a secure communication link.

2. The system of claim **1**, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim **2**, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim **3**, wherein the non-standard top-level domain name is one of .scom, .sorg, .snet, .sgov, .sedu, .smil and .sint.

5. The system of claim **2**, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

6. The system of claim **1**, wherein the communication network includes the Internet.

7. The system of claim **1**, wherein the domain name service system comprises an edge router.

8. The system of claim **1**, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim **8**, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

10. The system of claim **8**, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim **8**, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim **8**, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim **8**, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim **1**, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim **1**, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim **1**, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim **1**, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim **1**, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim **1**, wherein the domain name service system comprises a server.

20. The system of claim **19**, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim **1**, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim **1**, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.

23. The system of claim **1**, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim **1**, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim **1**, wherein at least one of the plurality of domain names comprises a secure name.

26. The system of claim **1**, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim **1**, wherein the secure communication link uses encryption.

29. The system of claim **1**, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim **29**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim **30**, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim **29**, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim **33**, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim **1**, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for:

connecting the domain name service system to a communication network;

storing a plurality of domain names and corresponding network addresses;

receiving a query for a network address; and

supporting an indication that the domain name service system supports establishing a secure communication link.

37. The machine-readable medium of claim **36**, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The machine-readable medium of claim **36**, wherein the instructions comprise code for responding to the query for the network address.

39. The machine-readable medium of claim **36**, wherein the instructions comprise code for providing, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The machine-readable medium of claim **36**, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The machine-readable medium of claim **36**, wherein the instructions comprise code for indicating that the domain name service system supports the establishment of a secure communication link.

42. The machine-readable medium of claim **36**, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The machiue-readable medium of claim **36**, wherein the code resides on a server.

44. The machine-readable medium of claim **36**, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The machine-readable medium of claim **36**, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The machine-readable medium of claim **36**, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The machine-readable medium of claim **36**, wherein the instructions comprise code for authenticating the query for the network address.

48. The machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes a secure name.

50. The machine-readable medium of claim **36**, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The machine-readable medium of claim **36**, wherein the secure communication link uses encryption.

53. The machine-readable medium of claim **36**, wherein the secure communication link is capable of supporting a plurality of services.

54. The machine-readable medium of claim **53**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The machine-readable medium of claim **54**, wherein the plurality of application programs comprises items selected from a group consisting of the following:

video conferencing, e-mail, a word processing program, and telephony.

56. The machine-readable medium of claim **53**, wherein the plurality of services comprises audio, video, or a combination thereof.

57. The machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The machine-readable medium of claim **57**, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

59. The machine-readable medium of claim **36**, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain name is response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

connecting a domain name service system to a communication network, the domain name service system comprising an indication that the domain name service system supports establishing a secure communication link;

storing a plurality of domain names and corresponding network addresses; and

receiving a query for a network address for communication.

* * * * *

TAB 8

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

March 30, 2011

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *7,490,151*
ISSUE DATE: *February 10, 2009*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. WALLACE
Certifying Officer

Plaintiffs' VirnetX Exhibit
VirnetX, Inc. v. Apple, Inc.
PX007
C.A. 6:10-cv-0417

US007490151B2

(12) **United States Patent**
Munger et al.

(10) **Patent No.: US 7,490,151 B2**
(45) **Date of Patent: Feb. 10, 2009**

(54) **ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST**

(75) Inventors: **Edward Colby Munger**, Crownsville, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **Virnetx Inc.**, Scotts Valley Drive, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 818 days.

(21) Appl. No.: **10/259,494**

(22) Filed: **Sep. 30, 2002**

(65) **Prior Publication Data**

US 2003/0037142 A1 Feb. 20, 2003

Related U.S. Application Data

(60) Division of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/137,704, filed on Jun. 7, 1999, provisional application No. 60/106,261, filed on Oct. 30, 1998.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/225**; 709/229

(58) **Field of Classification Search** 709/217-225, 709/229; 713/201

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,933,846	A	6/1990	Humphrey et al.

(Continued)

FOREIGN PATENT DOCUMENTS

DE	199 24 575	12/1999

(Continued)

OTHER PUBLICATIONS

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

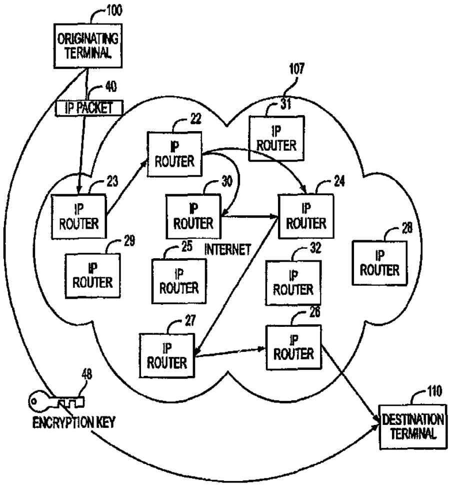
(Continued)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—McDermott Will & Emery

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

16 Claims, 35 Drawing Sheets



U.S. PATENT DOCUMENTS

4,988,990	A		1/1991	Warrior	
5,164,986	A	*	11/1992	Bright	380/273
5,276,735	A		1/1994	Boebert et al.	
5,311,593	A		5/1994	Carmi	
5,329,521	A		7/1994	Walsh et al.	
5,341,426	A		8/1994	Barney et al.	
5,367,643	A		11/1994	Chang et al.	
5,559,883	A		9/1996	Williams	
5,561,669	A		10/1996	Lenney et al.	
5,588,060	A		12/1996	Aziz	
5,625,626	A		4/1997	Umekita	
5,654,695	A		8/1997	Olnowich et al.	
5,682,480	A		10/1997	Nakagawa	
5,689,566	A		11/1997	Nguyen	
5,740,375	A		4/1998	Dunne et al.	
5,774,660	A		6/1998	Brendel et al.	
5,787,172	A		7/1998	Arnold	
5,790,548	A	*	8/1998	Sistanizadeh et al.	370/401
5,796,942	A		8/1998	Esbensen	
5,805,801	A		9/1998	Holloway et al.	
5,842,040	A		11/1998	Hughes et al.	
5,845,091	A		12/1998	Dunne et al.	
5,867,650	A		2/1999	Osterman	
5,870,610	A		2/1999	Beyda et al.	
5,878,231	A		3/1999	Baehr et al.	
5,892,903	A		4/1999	Klaus	
5,898,830	A	*	4/1999	Wesinger et al.	726/15
5,905,859	A		5/1999	Holloway et al.	
5,918,019	A		6/1999	Valencia	
5,996,016	A		11/1999	Thalheimer et al.	
6,006,259	A		12/1999	Adelman et al.	
6,006,272	A		12/1999	Aravamudan et al.	
6,016,318	A		1/2000	Tomoike	
6,016,512	A		1/2000	Huitema	
6,041,342	A		3/2000	Yamaguchi	
6,052,788	A		4/2000	Wesinger, Jr. et al.	
6,055,574	A		4/2000	Smorodinsky et al.	
6,061,736	A		5/2000	Rochberger et al.	
6,079,020	A	*	6/2000	Liu	713/201
6,092,200	A		7/2000	Muniyappa et al.	
6,101,182	A	*	8/2000	Sistanizadeh et al.	370/352
6,119,171	A		9/2000	Alkhatib	
6,119,234	A	*	9/2000	Aziz et al.	713/201
6,147,976	A		11/2000	Shand et al.	
6,157,957	A		12/2000	Berthaud	
6,158,011	A		12/2000	Chen et al.	
6,168,409	B1		1/2001	Fare	
6,175,867	B1		1/2001	Taghadoss	
6,178,409	B1		1/2001	Weber et al.	
6,178,505	B1		1/2001	Schneider et al.	
6,179,102	B1		1/2001	Weber et al.	
6,222,842	B1		4/2001	Sasyan et al.	
6,226,751	B1		5/2001	Arrow et al.	
6,233,618	B1		5/2001	Shannon	
6,243,360	B1		6/2001	Basilico	
6,243,749	B1		6/2001	Sitaraman et al.	
6,243,754	B1		6/2001	Guerin et al.	
6,256,671	B1	*	7/2001	Strentzsch et al.	709/227
6,263,445	B1		7/2001	Blumenau	
6,286,047	B1		9/2001	Ramanathan et al.	
6,301,223	B1		10/2001	Hrastar et al.	
6,308,274	B1		10/2001	Swift	
6,311,207	B1		10/2001	Mighdoll et al.	
6,324,161	B1		11/2001	Kirch	
6,330,562	B1		12/2001	Boden et al.	
6,332,158	B1	*	12/2001	Risley et al.	709/219
6,353,614	B1		3/2002	Borella et al.	
6,425,003	B1	*	7/2002	Herzog et al.	709/223
6,430,155	B1		8/2002	Davie et al.	
6,430,610	B1		8/2002	Carter	
6,487,598	B1		11/2002	Valencia	
6,502,135	B1	*	12/2002	Munger et al.	709/225
6,505,232	B1		1/2003	Mighdoll et al.	
6,510,154	B1		1/2003	Mayes et al.	
6,549,516	B1		4/2003	Albert et al.	
6,557,037	B1		4/2003	Provino	
6,571,296	B1		5/2003	Dillon	
6,571,338	B1		5/2003	Shaio et al.	
6,581,166	B1		6/2003	Hirst et al.	
6,606,708	B1	*	8/2003	Devine et al.	713/201
6,618,761	B2		9/2003	Munger et al.	
6,671,702	B2		12/2003	Kruglikov et al.	
6,687,551	B2		2/2004	Steindl	
6,714,970	B1		3/2004	Fiveash et al.	
6,717,949	B1		4/2004	Boden et al.	
6,751,738	B2	*	6/2004	Wesinger et al.	713/201
6,760,766	B1		7/2004	Sahlqvist	
6,826,616	B2		11/2004	Larson et al.	
6,839,759	B2		1/2005	Larson et al.	
7,010,604	B1		3/2006	Munger et al.	
7,133,930	B2		11/2006	Munger et al.	
7,188,180	B2		3/2007	Larson et al.	
7,197,563	B2		3/2007	Sheymov et al.	
2002/0004898	A1		1/2002	Droge	
2003/0196122	A1	*	10/2003	Wesinger et al.	713/201
2005/0055306	A1		3/2005	Miller et al.	
2006/0059337	A1	*	3/2006	Poyhonen et al.	713/165

FOREIGN PATENT DOCUMENTS

EP	0 814 589		12/1997
EP	0 814 589	A	12/1997
EP	0 838 930		4/1998
EP	0 838 930	A	4/1998
EP	836306	A1	4/1998
EP	0 858 189		8/1998
GB	2 317 792		4/1998
GB	2 317 792	A	4/1998
GB	2 334 181	A	8/1999
GB	2334181	A	8/1999
WO	9827783	A	6/1998
WO	WO 98/27783		6/1998
WO	WO 9827783	A	6/1998
WO	WO 98 55930		12/1998
WO	WO 98 59470		12/1998
WO	WO 99 38081		7/1999
WO	WO 99 48303		9/1999
WO	WO 00/17775		3/2000
WO	WO 00/70458		11/2000
WO	WO 01 50688		7/2001

OTHER PUBLICATIONS

Donald E. Eastlake, 3rd, "Domain Name System Security Extensions", Internet Draft, Apr. 1998, pp. 1-51.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-375.

P. Srisuresh et al., "DNA extensions to Network address Translators (DNS_ALG)", Internet Draft, Jul. 1998, pp. 1-27.

James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

D. Clark, "US Calls for Private Domain-Name System", Computer Society, Aug. 1, 1998, pp. 22-25.

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.

Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group, Apr. 1998, 51 pages.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-297 and pp. 351-375.

P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.

Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.

W. Stallings, "Cryptography And Network Security", 2nd Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.

Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.

Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE Infocom, 1996, pp. 1028-1036.

Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.

Laurie Wells (Lancasterbibelmail MSN COM); "Subject: Security Icon" Usenet Newsgroup, Oct. 19, 1998, XP002200606.

Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW '99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf> (Abstract).

Alan 0. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ssl13/ draft302.txt on Feb. 4, 2002, 56 pages.

Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: http://www. springerlink.com/content/4uac0tb0hecoma89/fulltext.pdf>.

Dolev, Shlomi and Ostrovsky, Rafil, Efficient Anonymous Multicast and Reception (Extended Abstract), 16 pages.

F. Halsall, "Data Communications, Computer Networks and Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.

Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/ doc/glossary.html on Feb. 21, 2002, 25 pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3.doc/rationale.html on Feb. 21, 2002, 4 pages.

Linux FreeS/WAN Index File, printed from http://liberty.freewan.org/freeswan trees/freeswan-1.3/doc/ on Feb. 21, 2002, 3 pages.

Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), Crowds: Anonymity for Web Transactions, pp. 1-23.

RFC 2401-Security Architecture for the Internet Protocol (RTP).

RFC 2543-SIP: Session Initiation Protocol (SIP or SIPS).

Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.

Search Report, IPER (dataed Nov. 13, 2002), International Application No. PCT/US01/04340.

Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.

Shankar, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY,NY 1986.

* cited by examiner

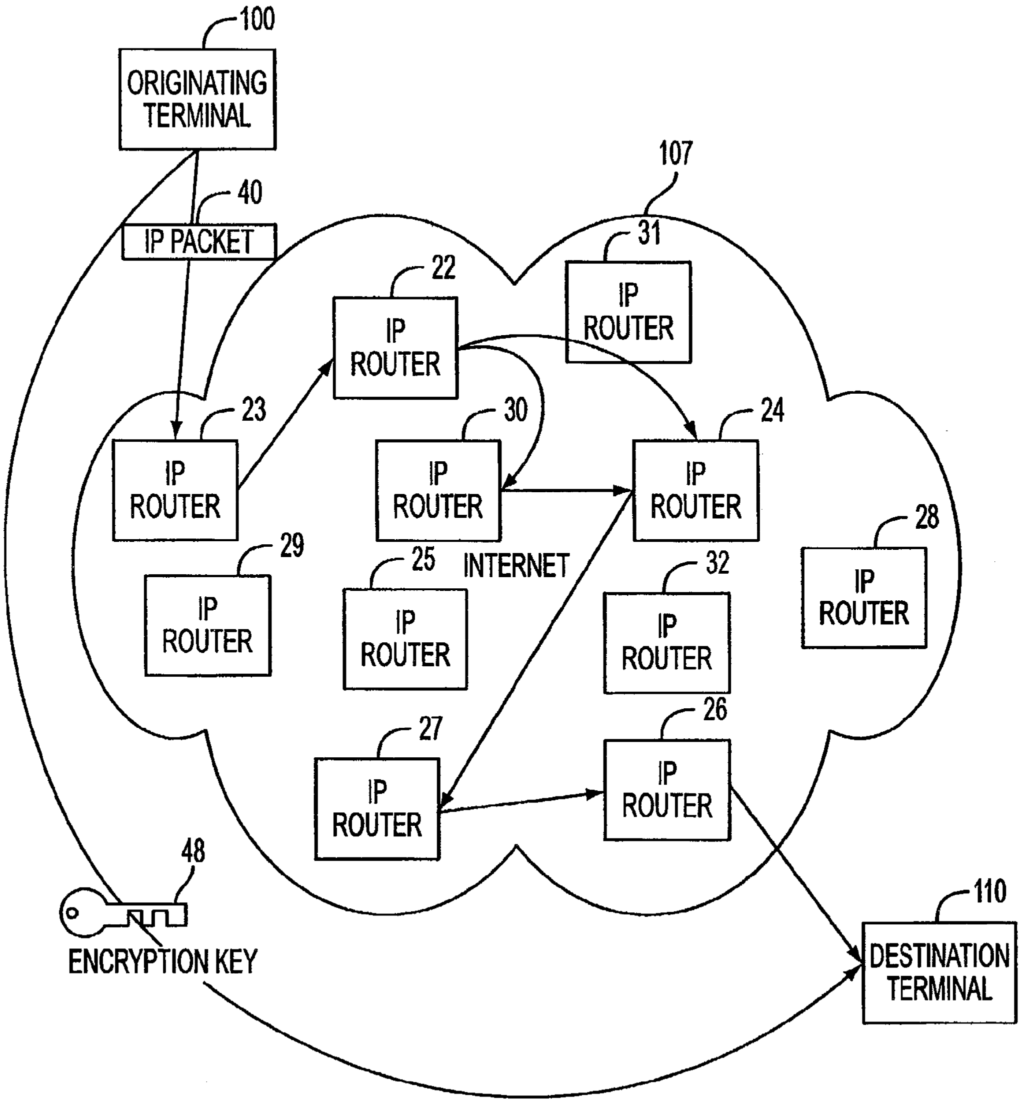

100
ORIGINATING TERMINAL
40
IP PACKET
107
31
IP ROUTER
22
IP ROUTER
23
IP ROUTER
30
IP ROUTER
24
IP ROUTER
28
IP ROUTER
29
IP ROUTER
25
IP ROUTER
INTERNET
32
IP ROUTER
27
IP ROUTER
26
IP ROUTER
48
ENCRYPTION KEY
110
DESTINATION TERMINAL

FIG. 1

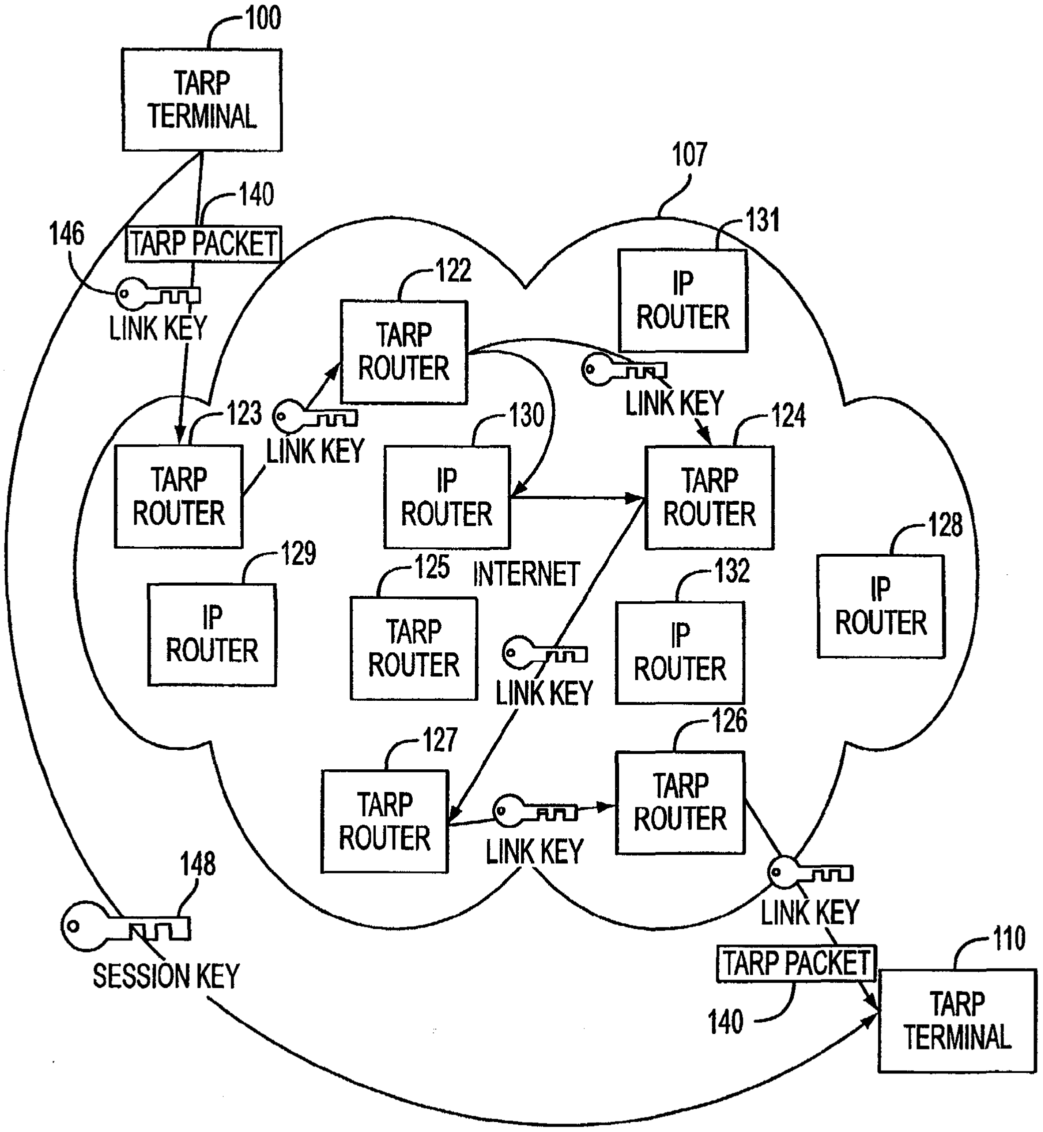

100
TARP TERMINAL
107
140
TARP PACKET
146
LINK KEY
131
IP ROUTER
122
TARP ROUTER
LINK KEY
123
LINK KEY
130
124
TARP ROUTER
IP ROUTER
TARP ROUTER
128
129
125
INTERNET
132
IP ROUTER
IP ROUTER
TARP ROUTER
IP ROUTER
LINK KEY
127
126
TARP ROUTER
TARP ROUTER
LINK KEY
148
LINK KEY
SESSION KEY
TARP PACKET
110
TARP TERMINAL
140

FIG. 2

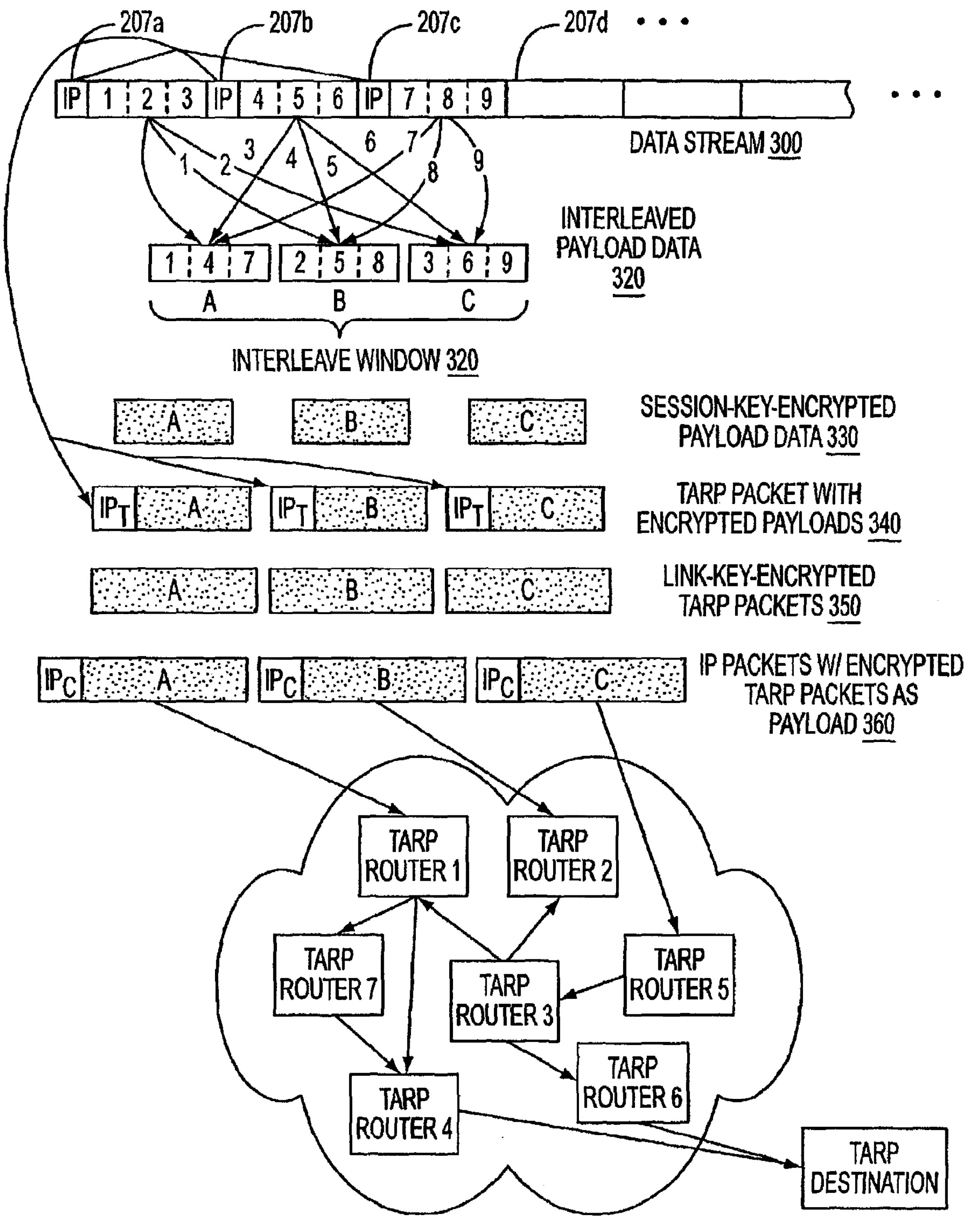

207a
207b
207c
207d
• • •
IP 1 2 3 IP 4 5 6 IP 7 8 9
• • •
DATA STREAM 300
1 2 3 4 5 6 7 8 9
INTERLEAVED PAYLOAD DATA 320
1 4 7
2 5 8
3 6 9
A B C
INTERLEAVE WINDOW 320
A B C
SESSION-KEY-ENCRYPTED PAYLOAD DATA 330
IP_T A
IP_T B
IP_T C
TARP PACKET WITH ENCRYPTED PAYLOADS 340
A B C
LINK-KEY-ENCRYPTED TARP PACKETS 350
IP_C A
IP_C B
IP_C C
IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD 360
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

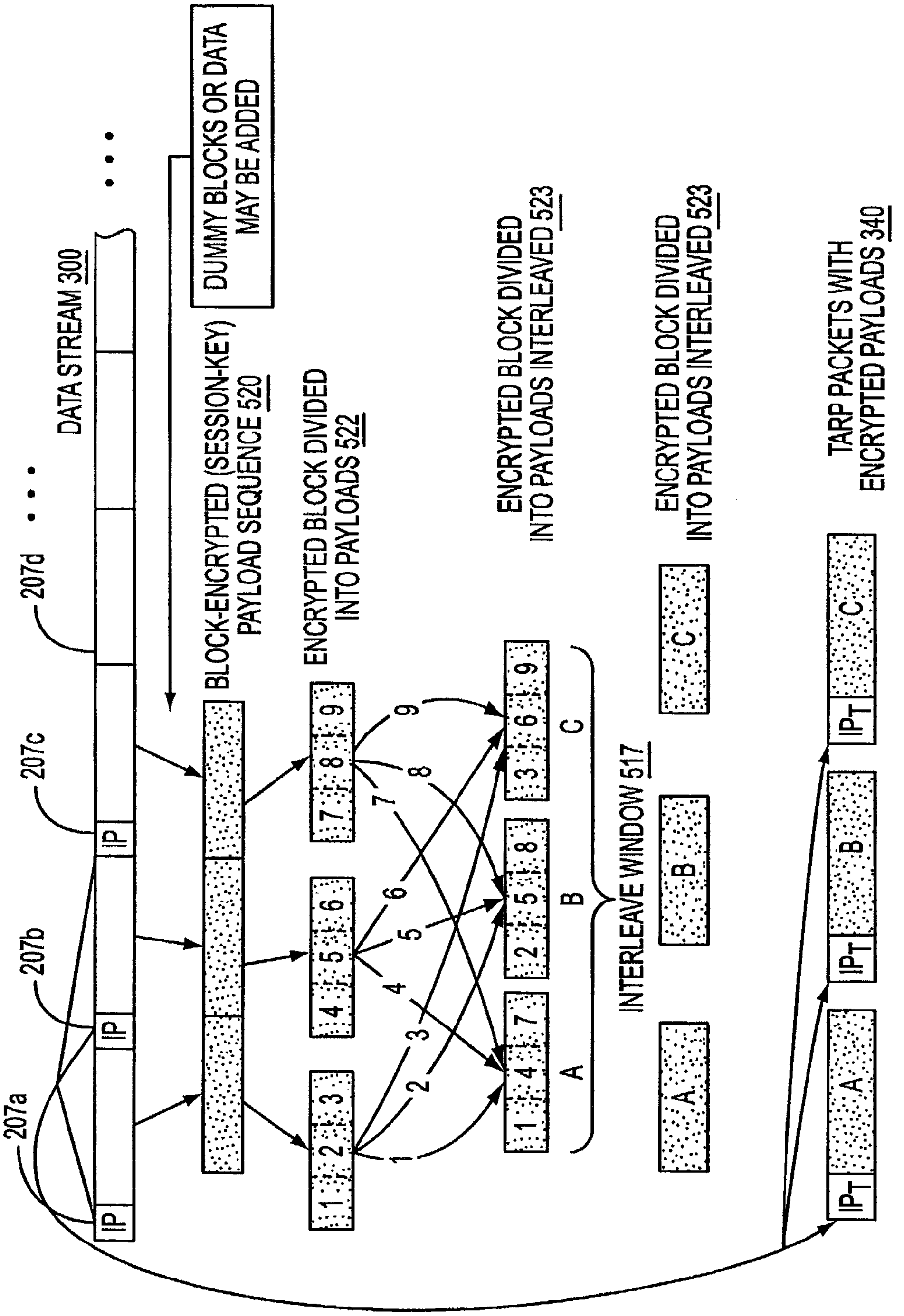

DATA STREAM 300
207a
207b
207c
207d
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE 520
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS 522
1 2 3
4 5 6
7 8 9
1 2 3 4 5 6 7 8 9
1 4 7
2 5 8
3 6 9
A
B
C
INTERLEAVE WINDOW 517
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
A
B
C
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
IP_T
A
IP_T
B
IP_T
C
TARP PACKETS WITH ENCRYPTED PAYLOADS 340

FIG. 3B

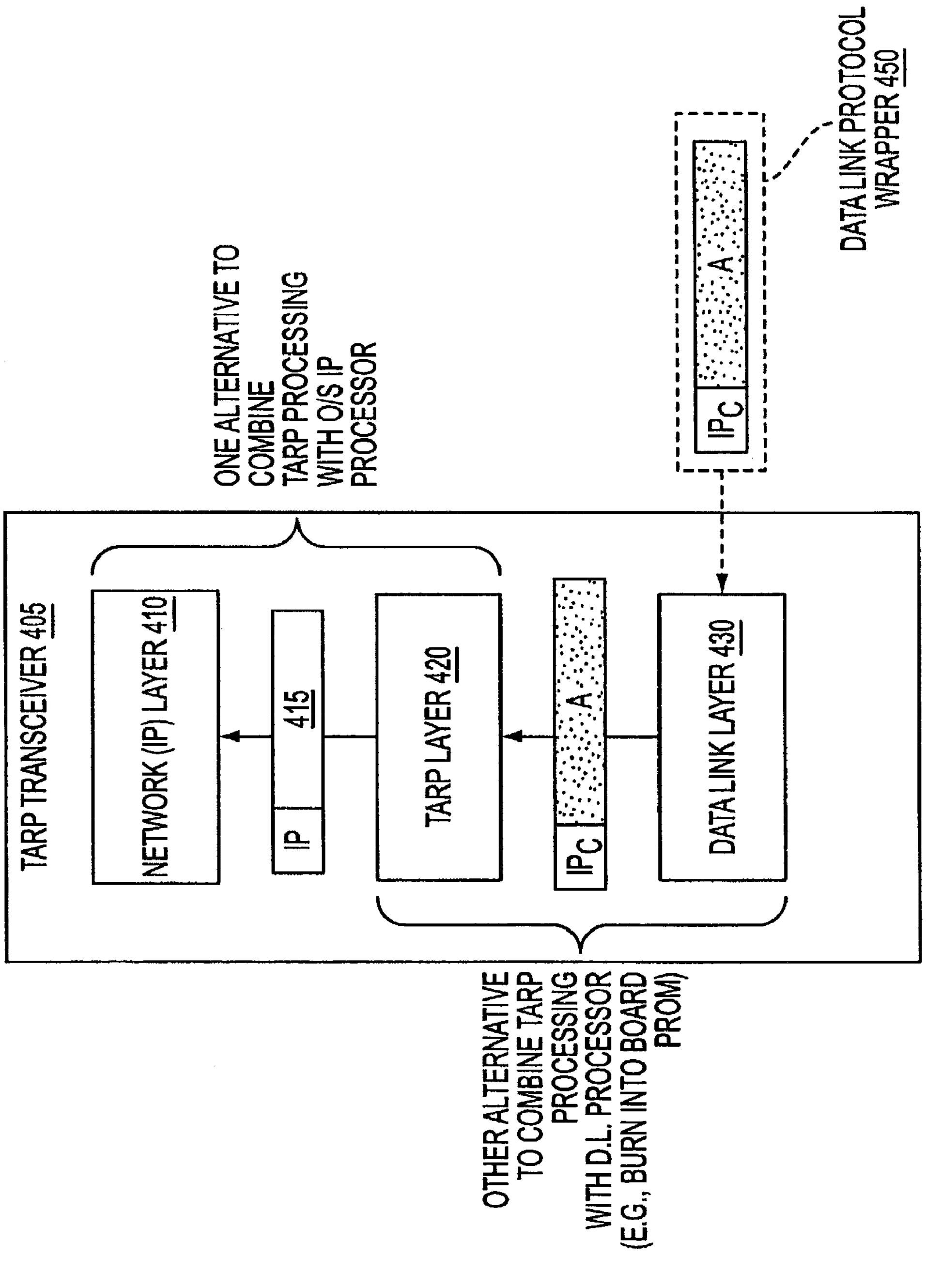

DATA LINK PROTOCOL WRAPPER 450
IP_C
A
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
TARP TRANSCEIVER 405
NETWORK (IP) LAYER 410
IP
415
TARP LAYER 420
IP_C
A
DATA LINK LAYER 430
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (E.G., BURN INTO BOARD PROM)

FIG. 4

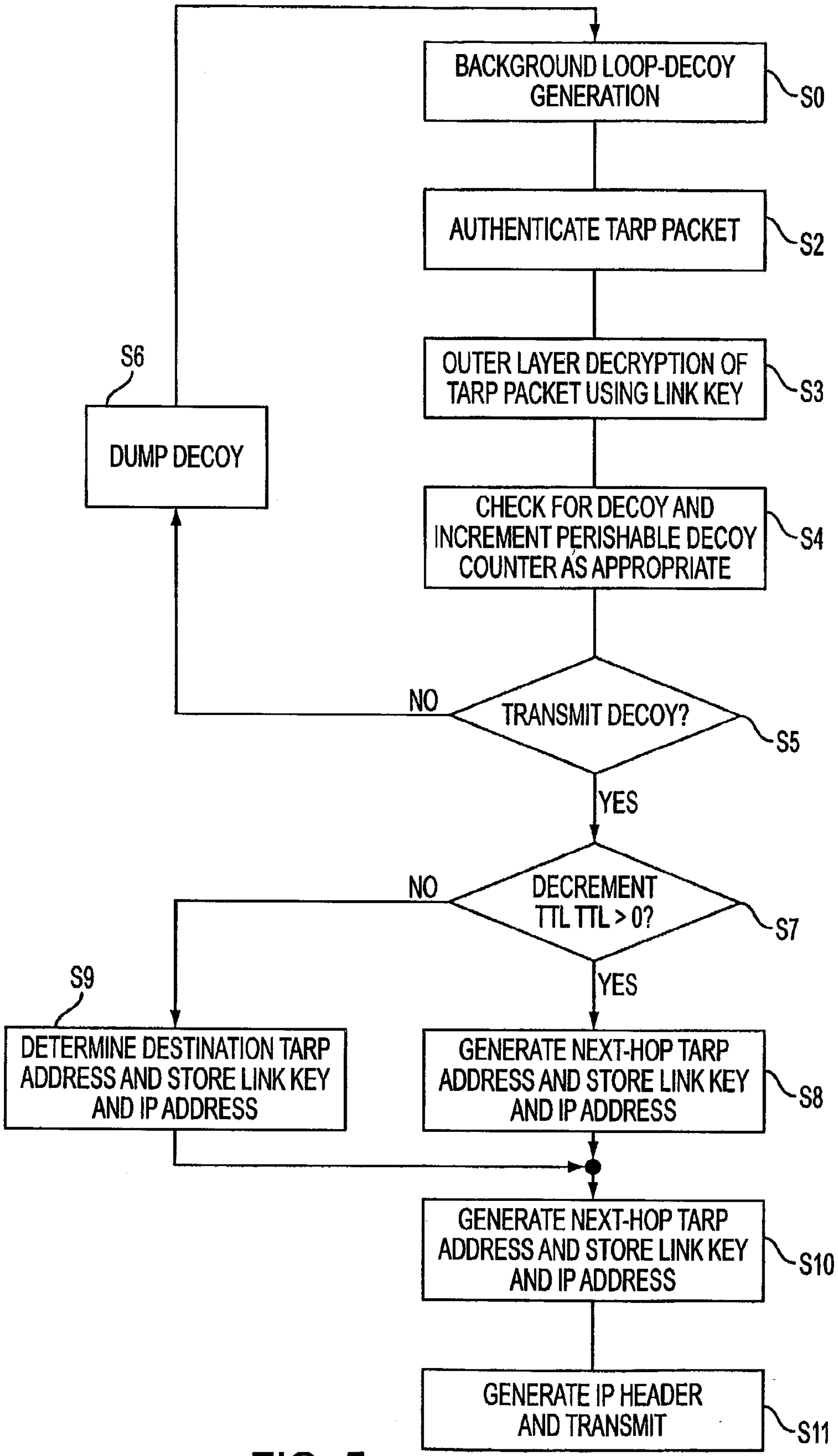

BACKGROUND LOOP-DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
S6
DUMP DECOY
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

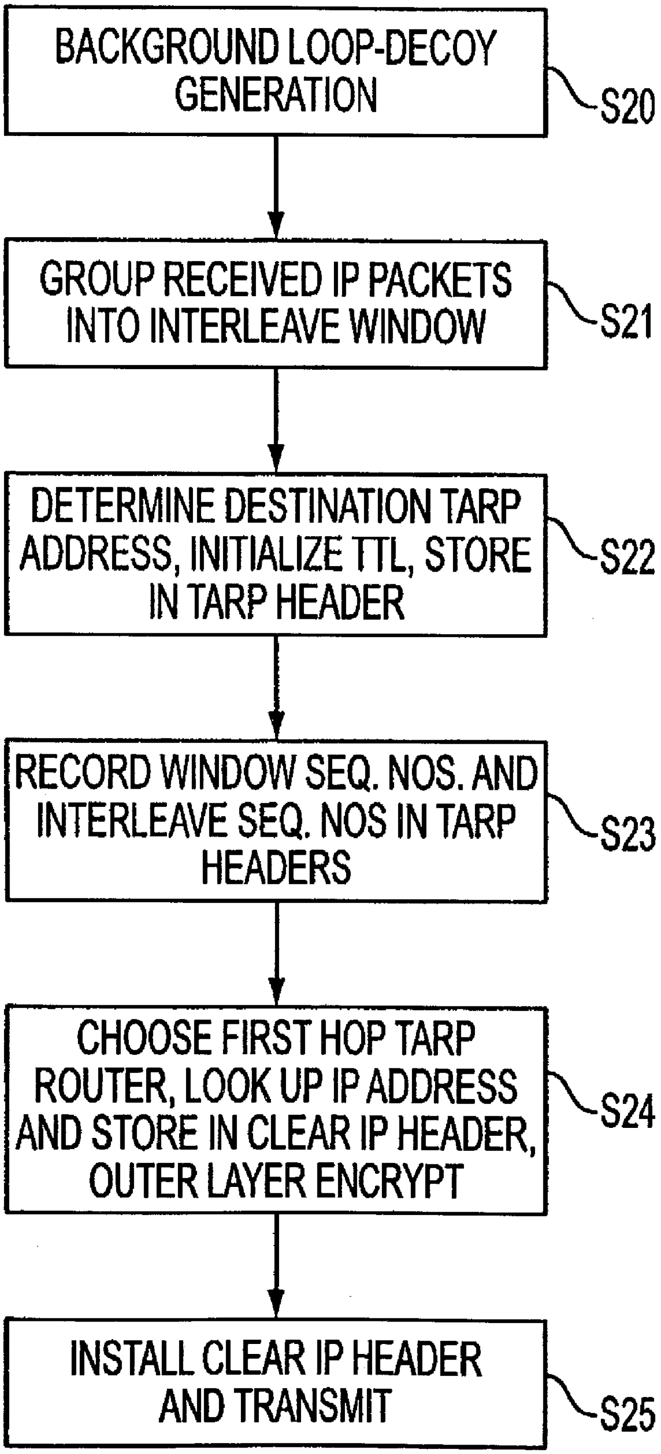

BACKGROUND LOOP-DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

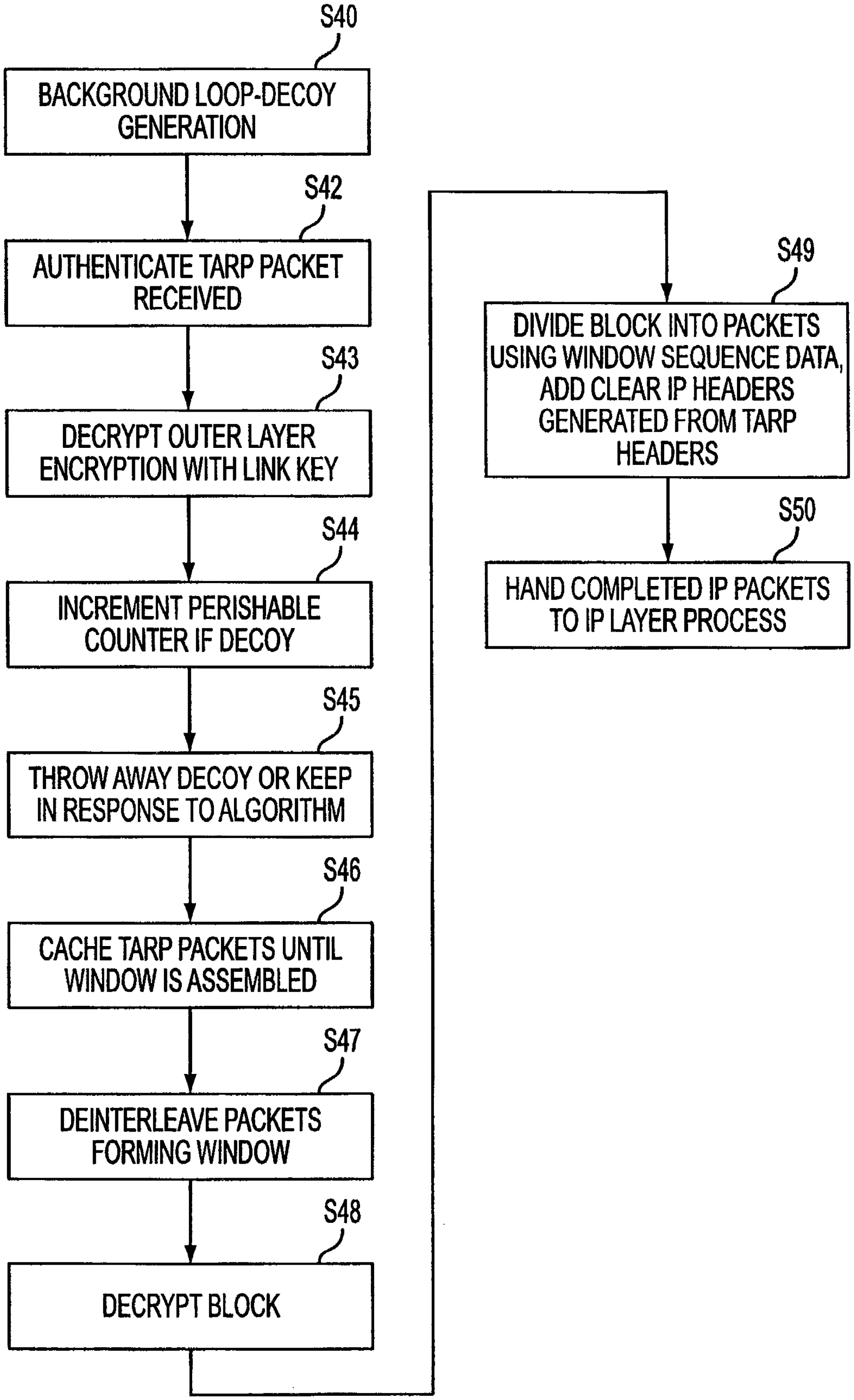

S40
BACKGROUND LOOP-DECOY GENERATION
S42
AUTHENTICATE TARP PACKET RECEIVED
S43
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S44
INCREMENT PERISHABLE COUNTER IF DECOY
S45
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S46
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S47
DEINTERLEAVE PACKETS FORMING WINDOW
S48
DECRYPT BLOCK
S49
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S50
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS

FIG. 7

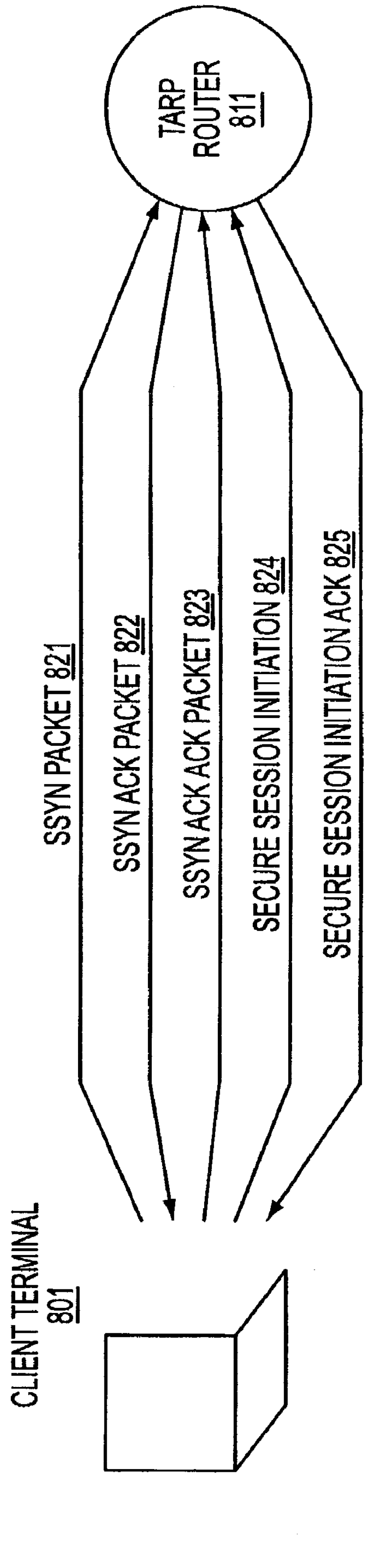
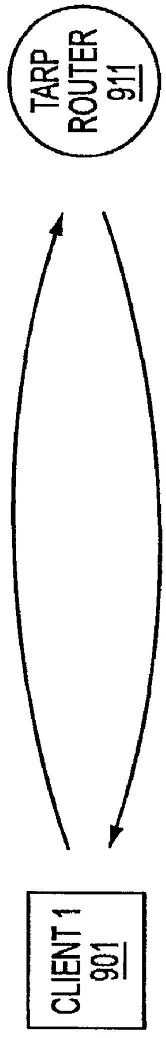

TARP ROUTER 811
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
SECURE SESSION INITIATION 824
SECURE SESSION INITIATION ACK 825
CLIENT TERMINAL 801

FIG. 8



TRANSMIT TABLE 921	
131.218.204.98	131.218.204.65
131.218.204.221	131.218.204.97
131.218.204.139	131.218.204.186
131.218.204.12	131.218.204.55
. . .	. . .

RECEIVE TABLE 922	
131.218.204.161	131.218.204.89
131.218.204.66	131.218.204.212
131.218.204.201	131.218.204.127
131.218.204.119	131.218.204.49
. . .	. . .

RECEIVE TABLE 924	
131.218.204.98	131.218.204.65
131.218.204.221	131.218.204.97
131.218.204.139	131.218.204.186
131.218.204.12	131.218.204.55
. . .	. . .

TRANSMIT TABLE 923	
131.218.204.161	131.218.204.89
131.218.204.66	131.218.204.212
131.218.204.201	131.218.204.127
131.218.204.119	131.218.204.49
. . .	. . .

FIG. 9

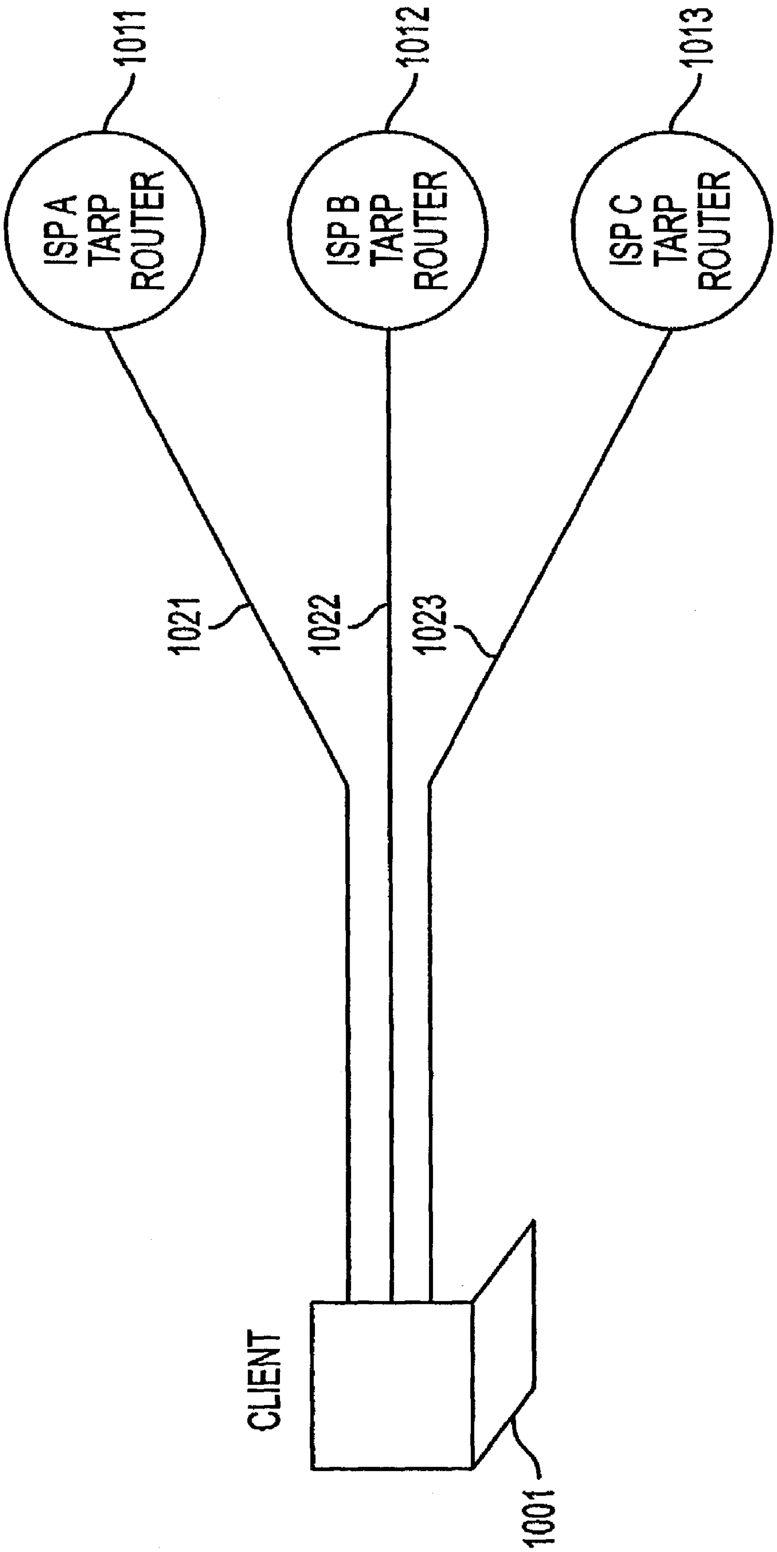

1011
1012
1013
ISP A TARP ROUTER
ISP B TARP ROUTER
ISP C TARP ROUTER
1021
1022
1023
CLIENT
1001

FIG. 10

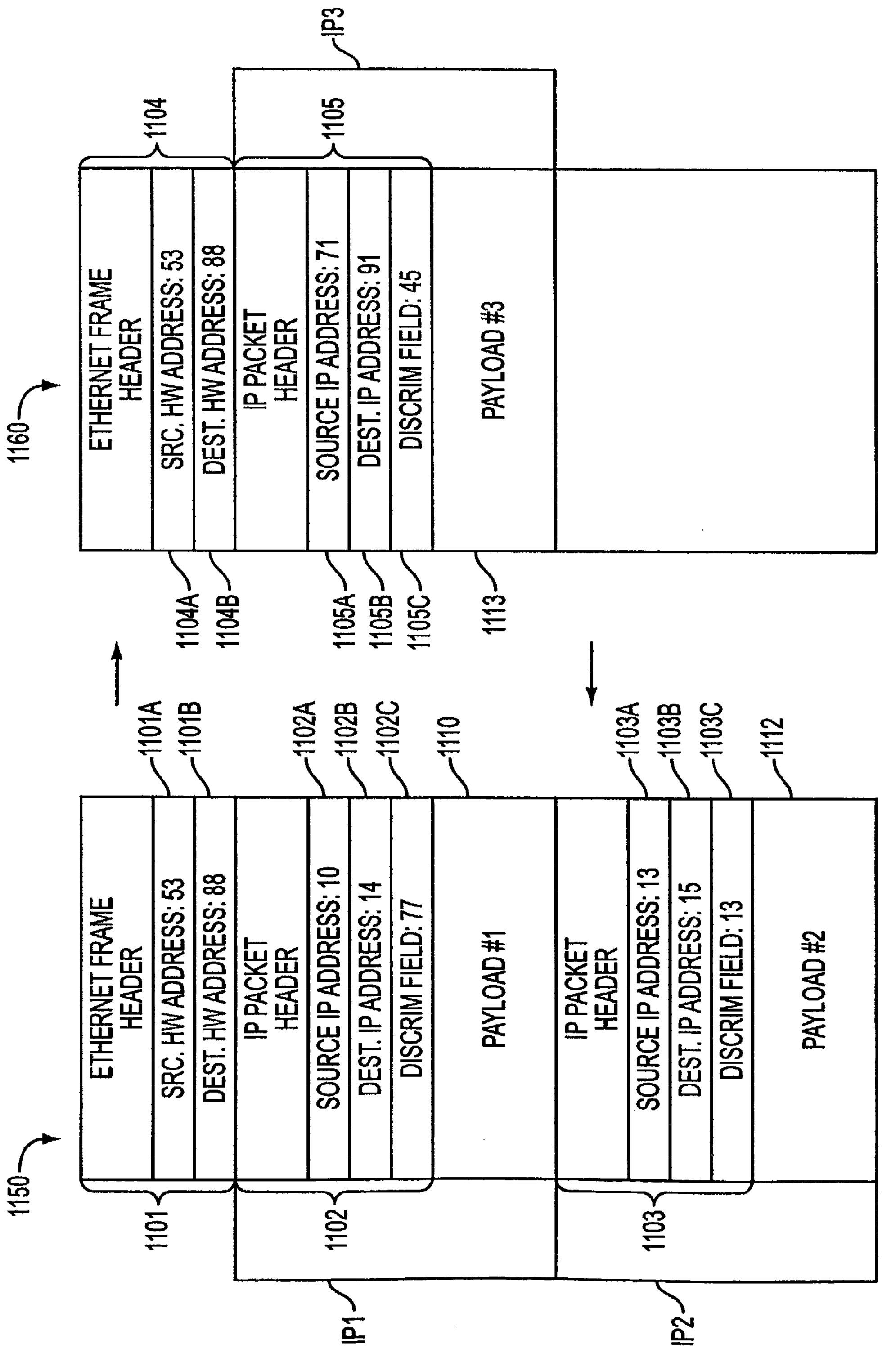
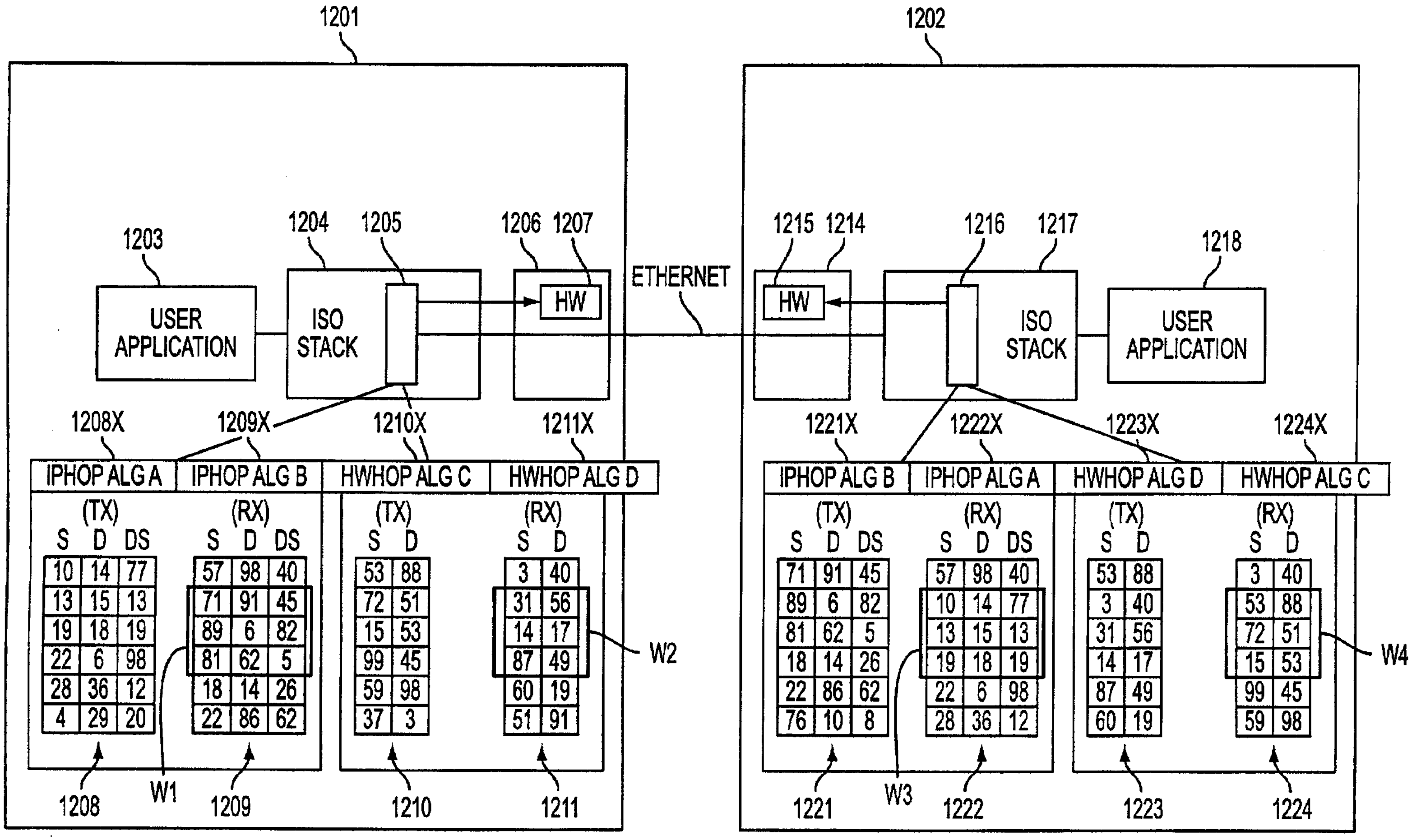

1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104
1105
IP3
1104A
1104B
1105A
1105B
1105C
1113
1150
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1101
1102
1103
IP1
IP2

FIG. 11



IPHOP ALG A (TX)

S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

IPHOP ALG B (RX)

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

HWHOP ALG C (TX)

S	D
53	88
72	51
15	53
99	45
59	98
37	3

HWHOP ALG D (RX)

S	D
3	40
31	56
14	17
87	49
60	19
51	91

IPHOP ALG B (TX)

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

IPHOP ALG A (RX)

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

HWHOP ALG D (TX)

S	D
53	88
3	40
31	56
14	17
87	49
60	19

HWHOP ALG C (RX)

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

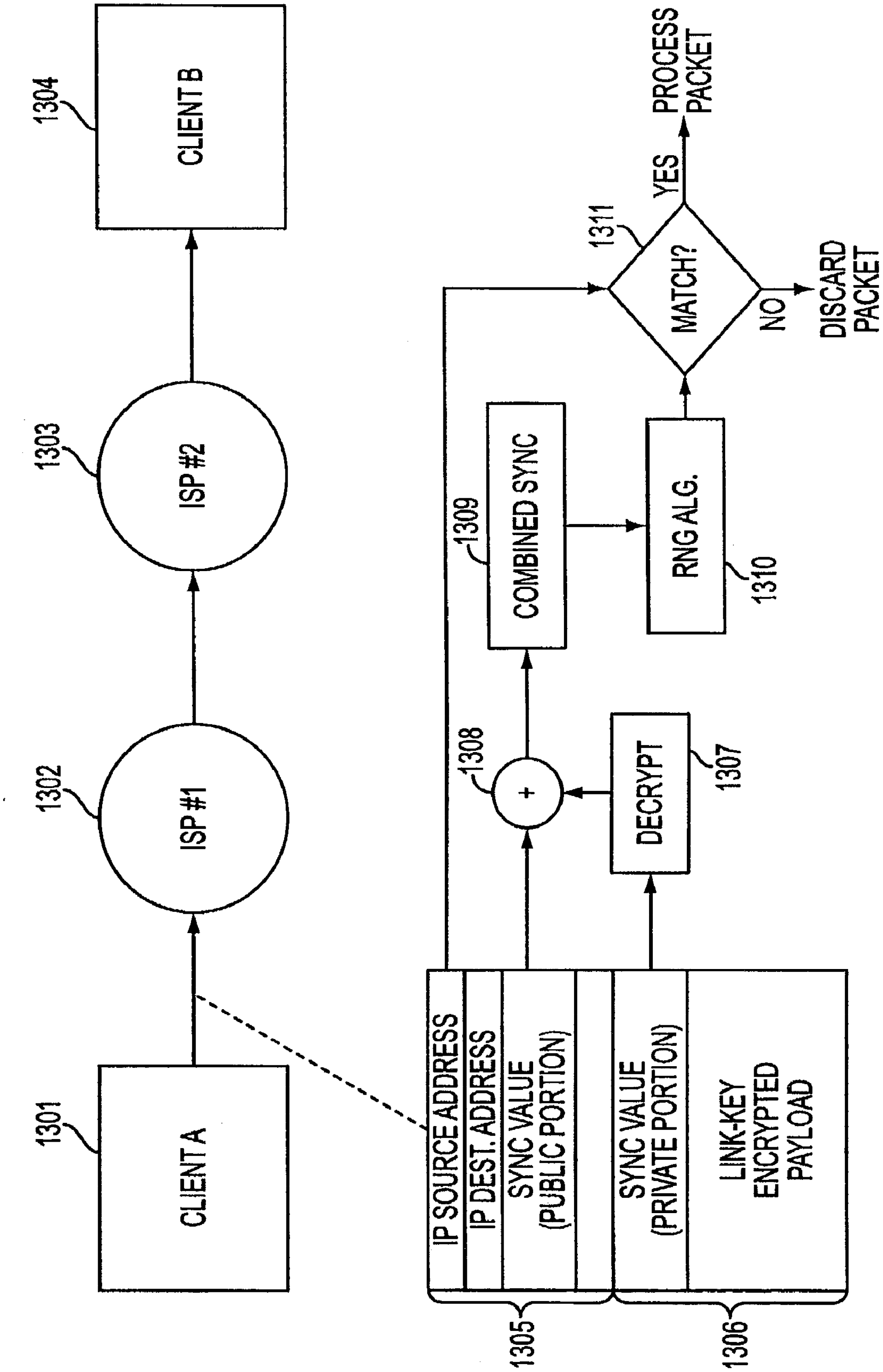

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
+
1308
COMBINED SYNC
1309
RNG ALG.
1310
MATCH?
1311
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

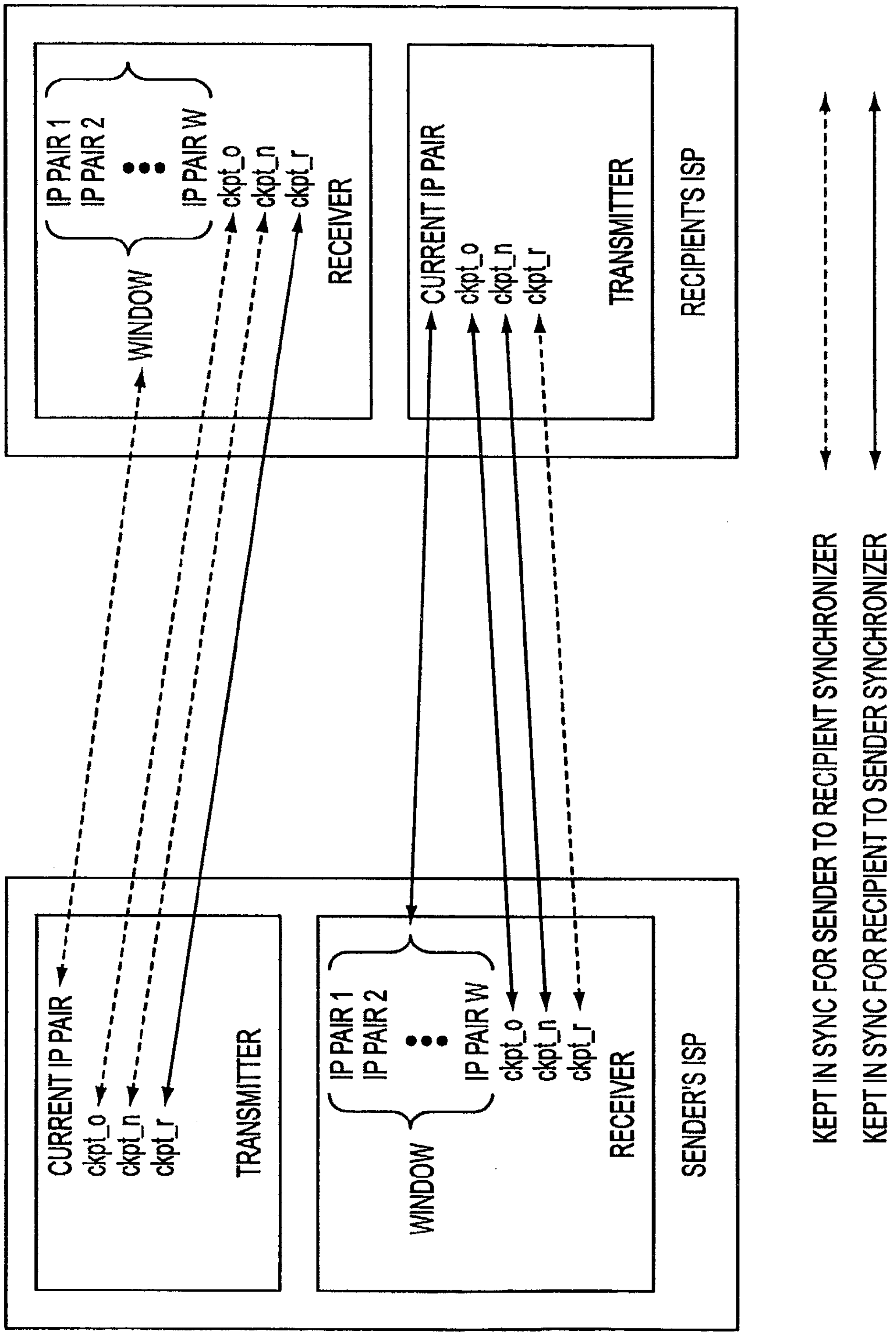

SENDER'S ISP
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
•••
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECIPIENT'S ISP
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
•••
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

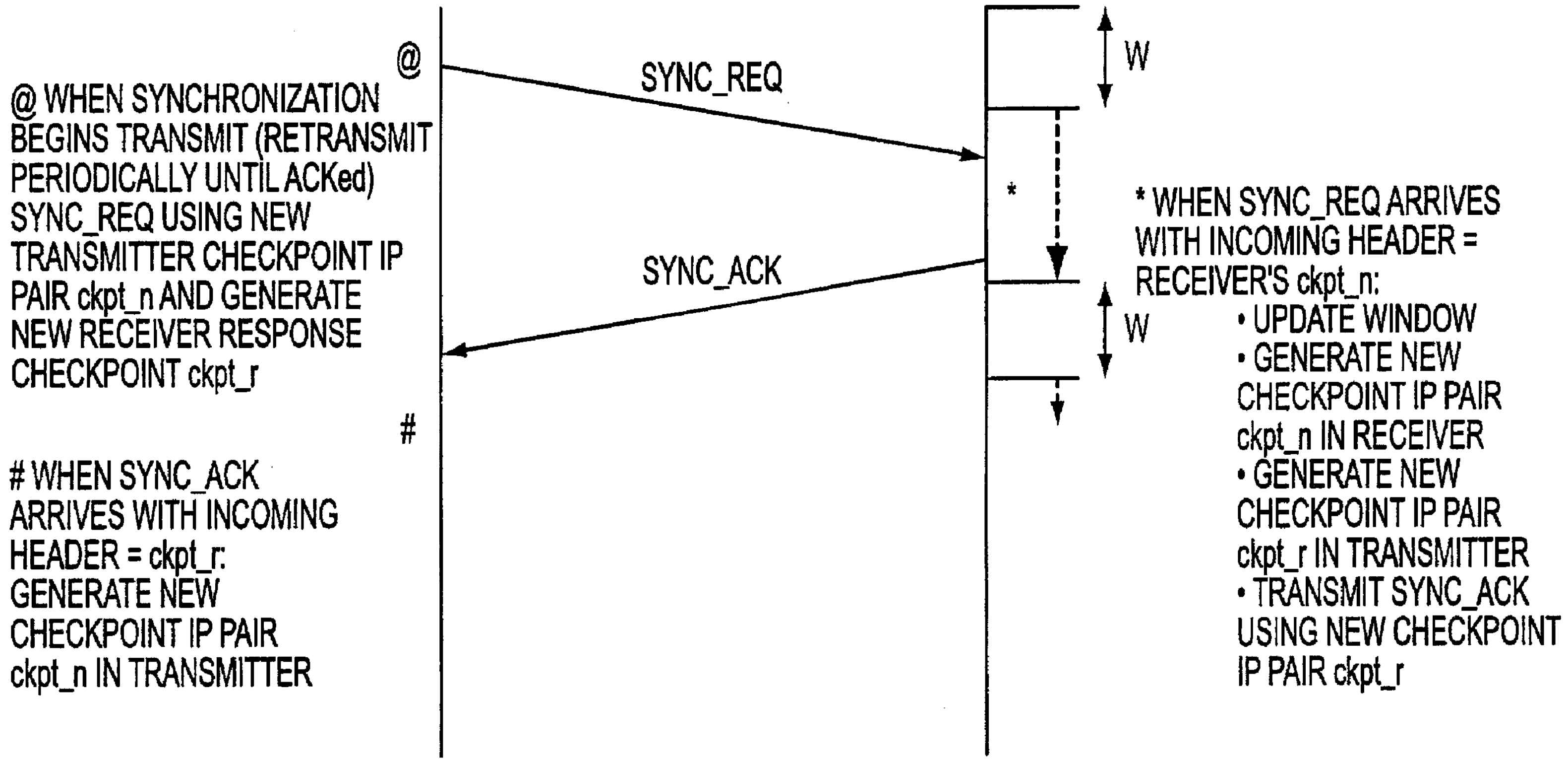

@
@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
#
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER
SYNC_REQ
SYNC_ACK
W
*
W
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
• UPDATE WINDOW
• GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
• GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
• TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r

FIG. 15

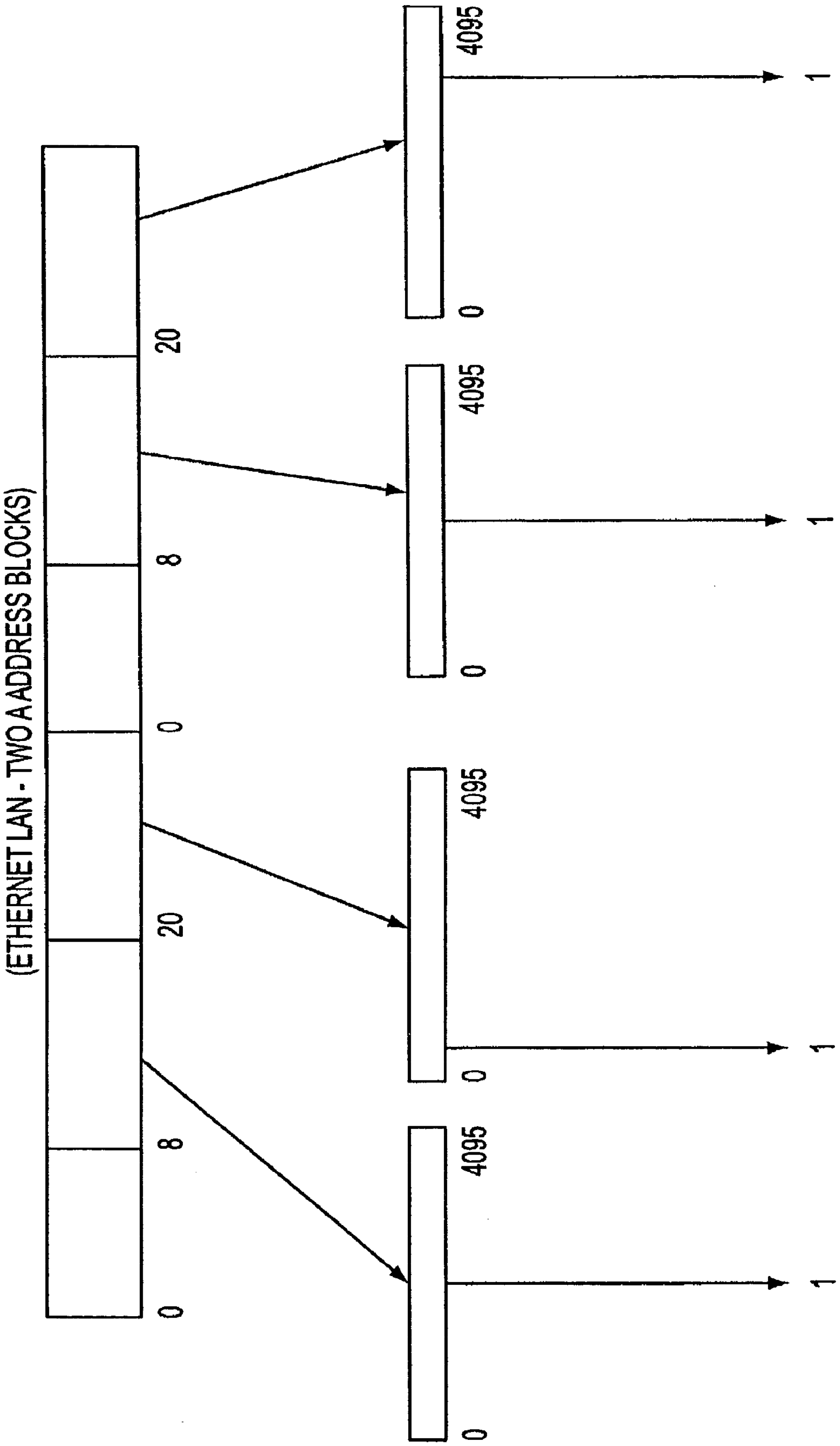

(ETHERNET LAN - TWO A ADDRESS BLOCKS)
0
8
20
0
8
20
0
4095
1
0
4095
1
0
4095
1
0
4095
1

FIG. 16

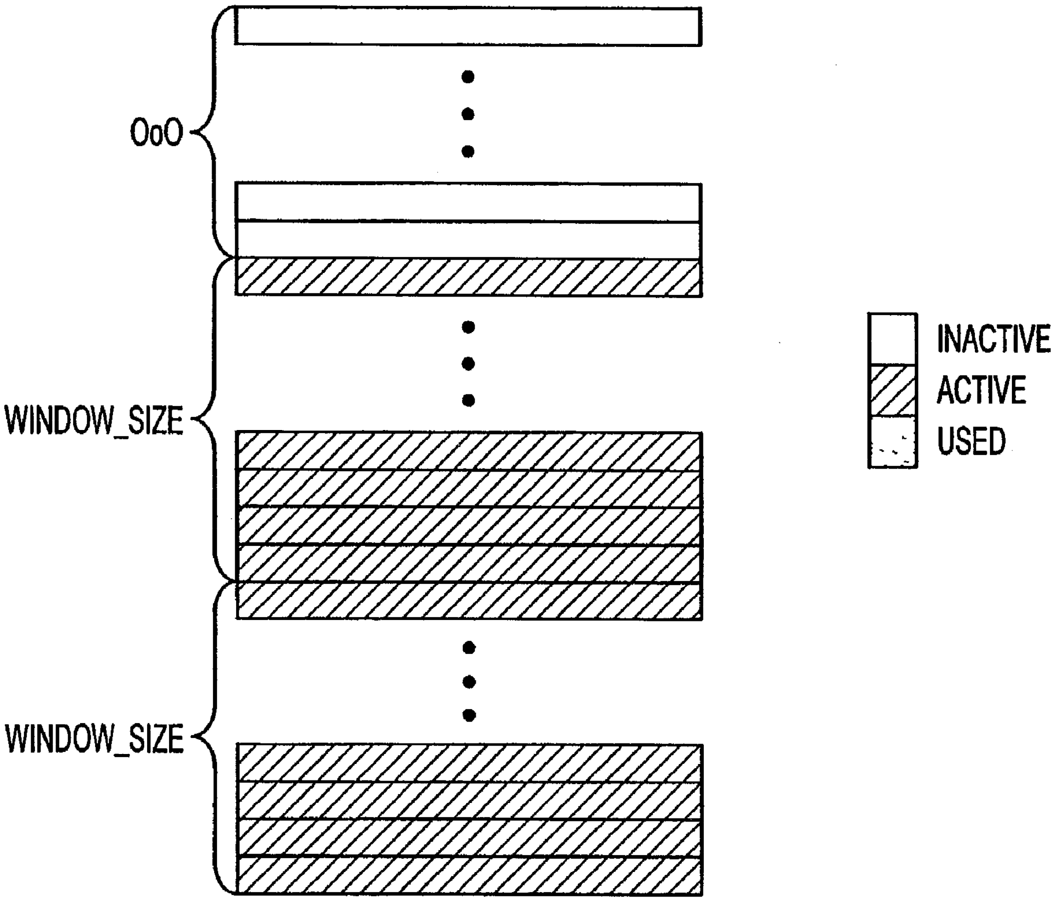


FIG. 17

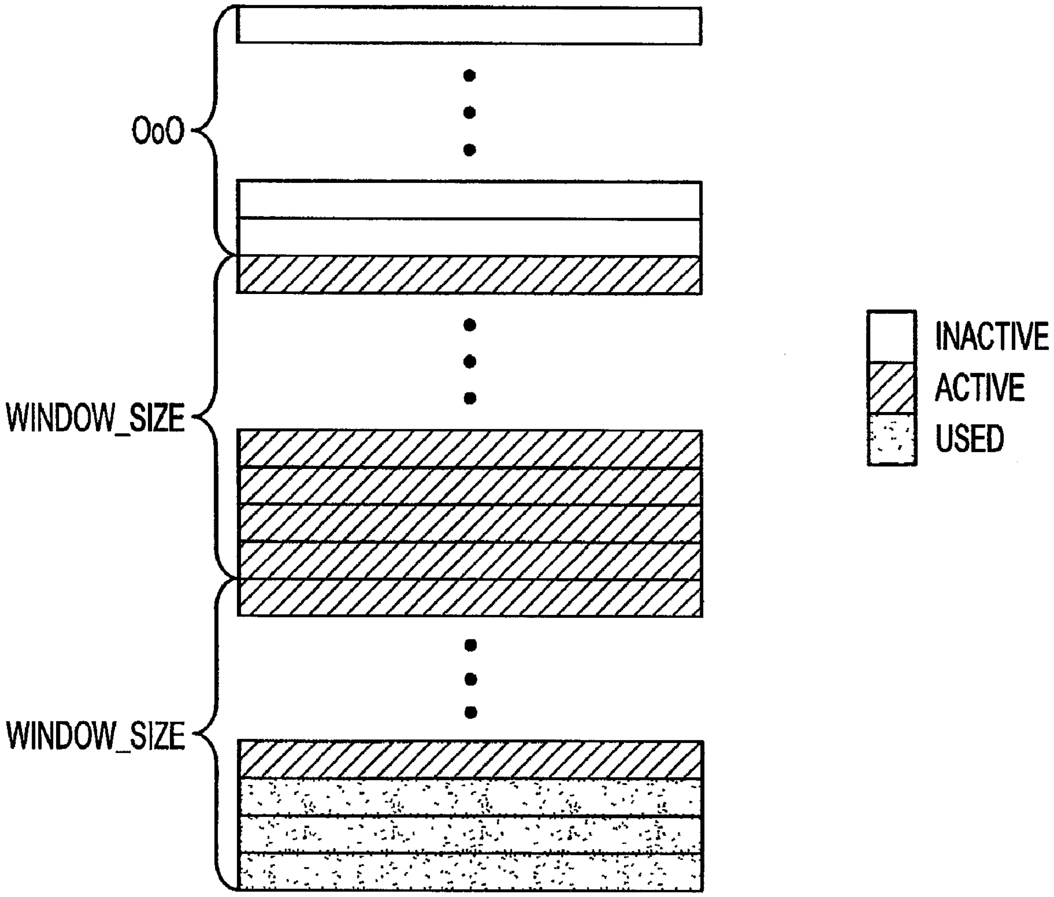


FIG. 18

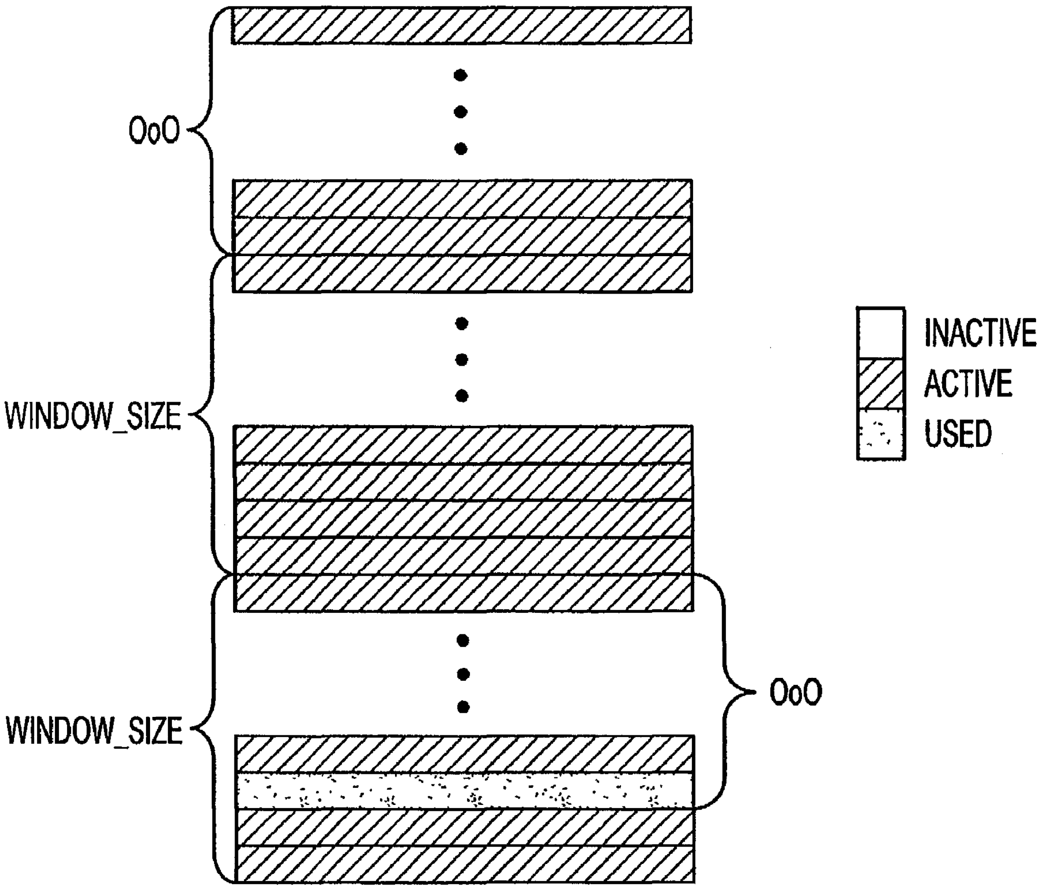


FIG. 19

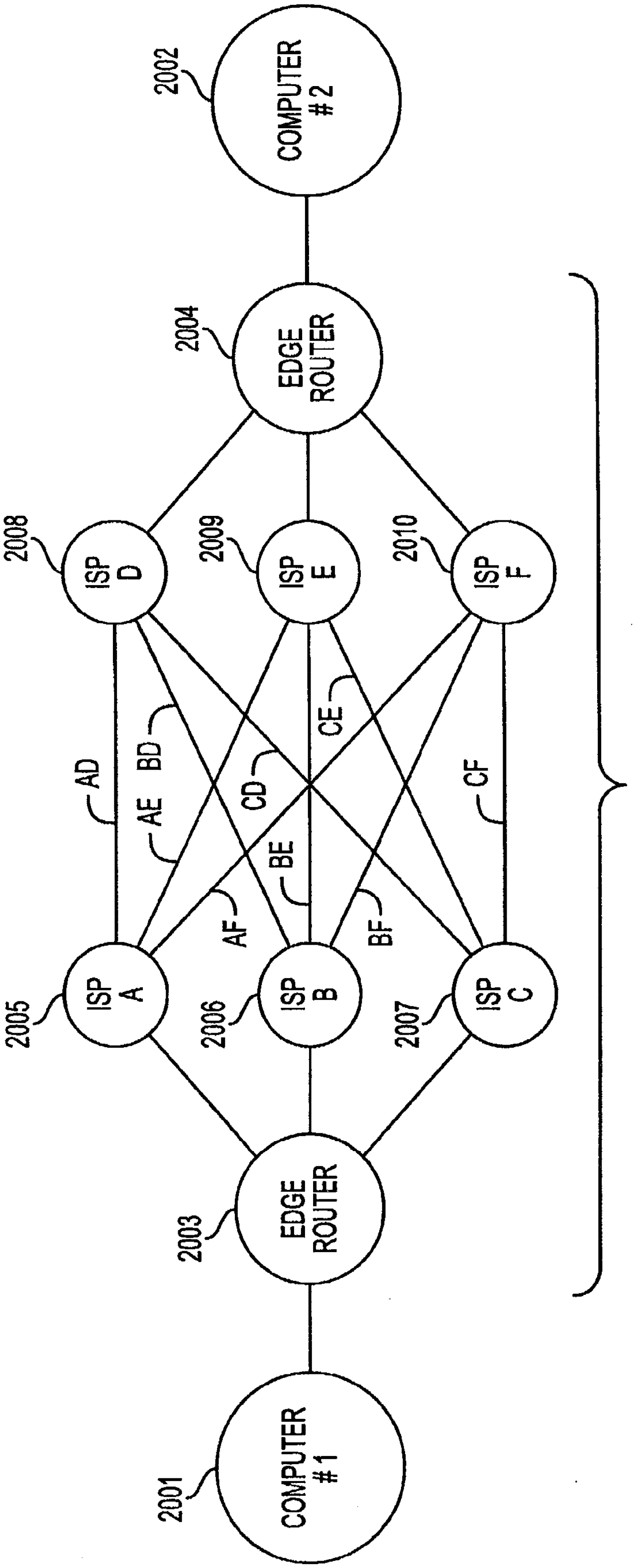

2002
COMPUTER #2
2004
EDGE ROUTER
2008
ISP D
2009
ISP E
2010
ISP F
AD
AE
BD
CD
CE
BE
AF
BF
CF
2011
2005
ISP A
2006
ISP B
2007
ISP C
2003
EDGE ROUTER
2001
COMPUTER #1

FIG. 20

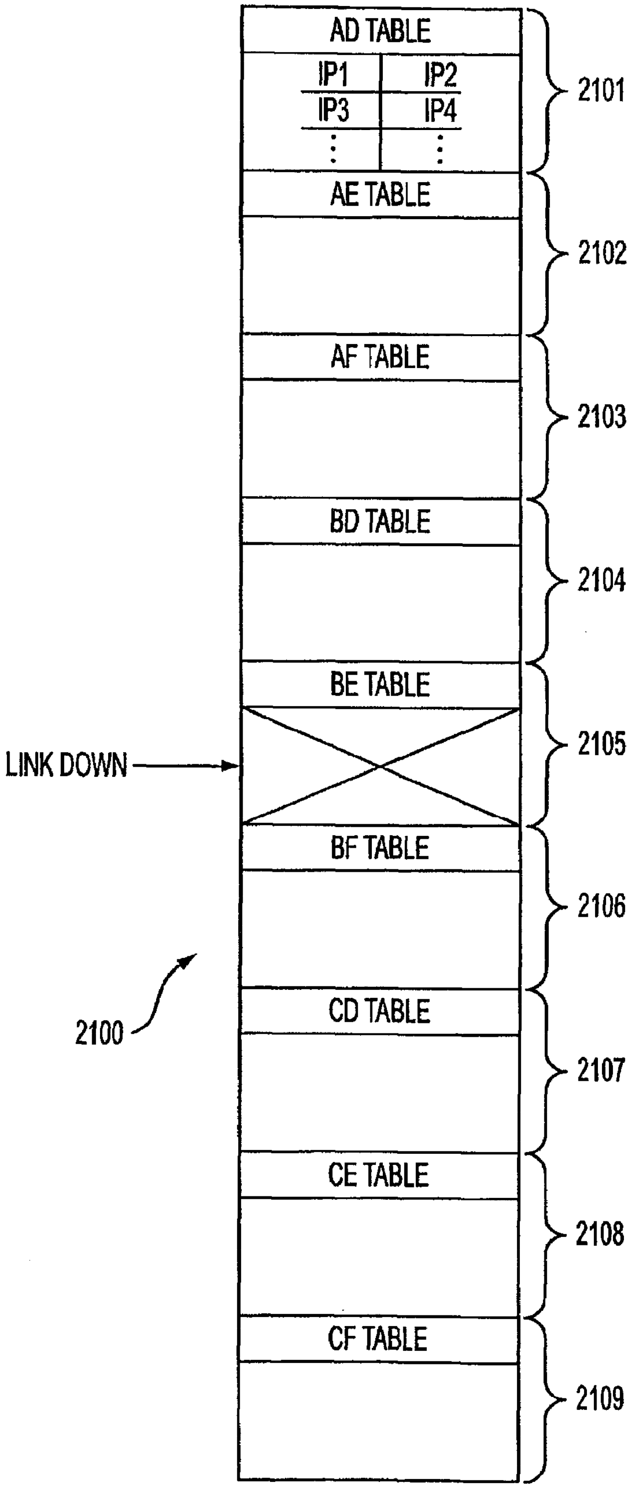

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

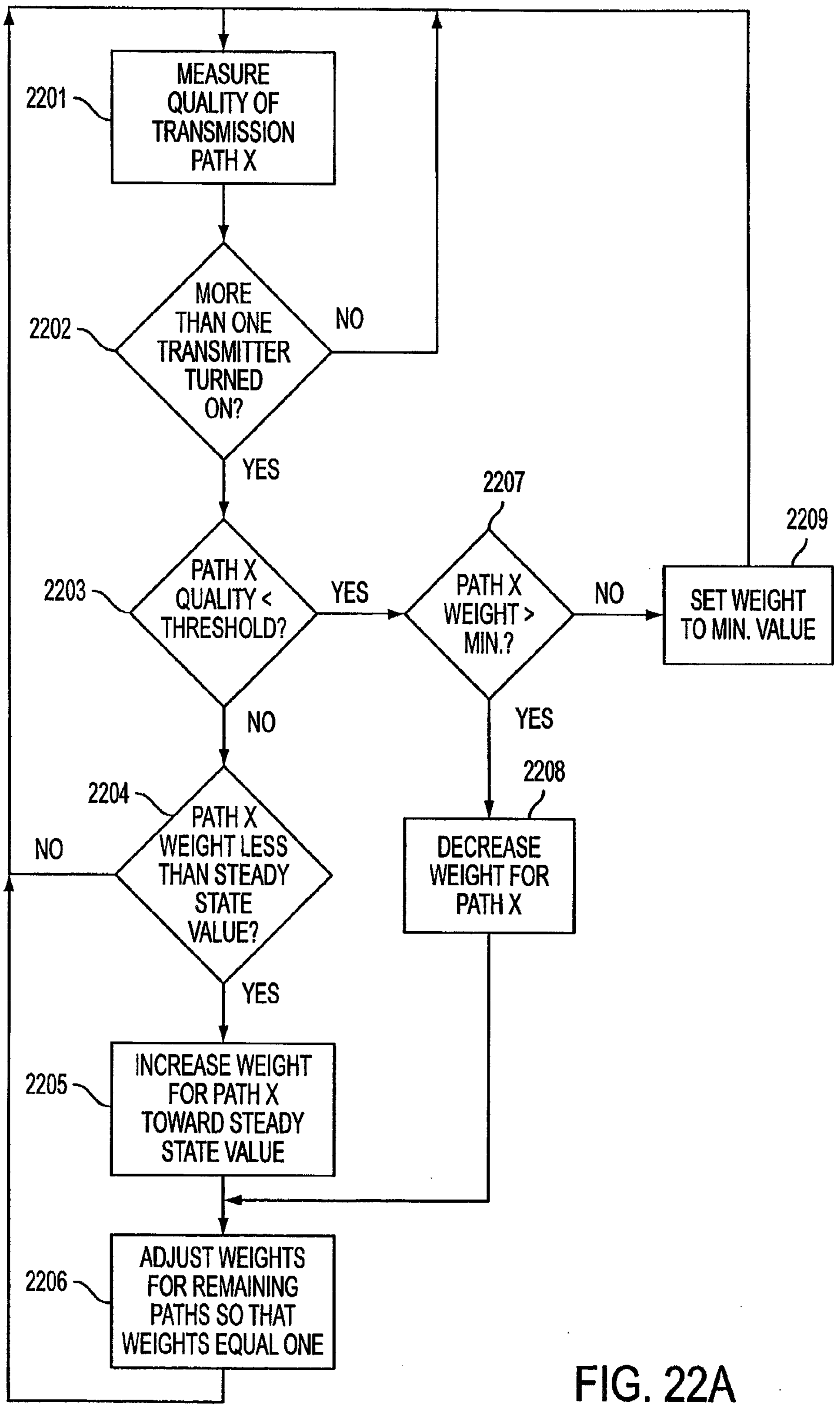

MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
2202
NO
YES
2207
2209
PATH X QUALITY < THRESHOLD?
2203
YES
PATH X WEIGHT > MIN.?
NO
SET WEIGHT TO MIN. VALUE
NO
YES
2208
2204
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
DECREASE WEIGHT FOR PATH X
YES
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

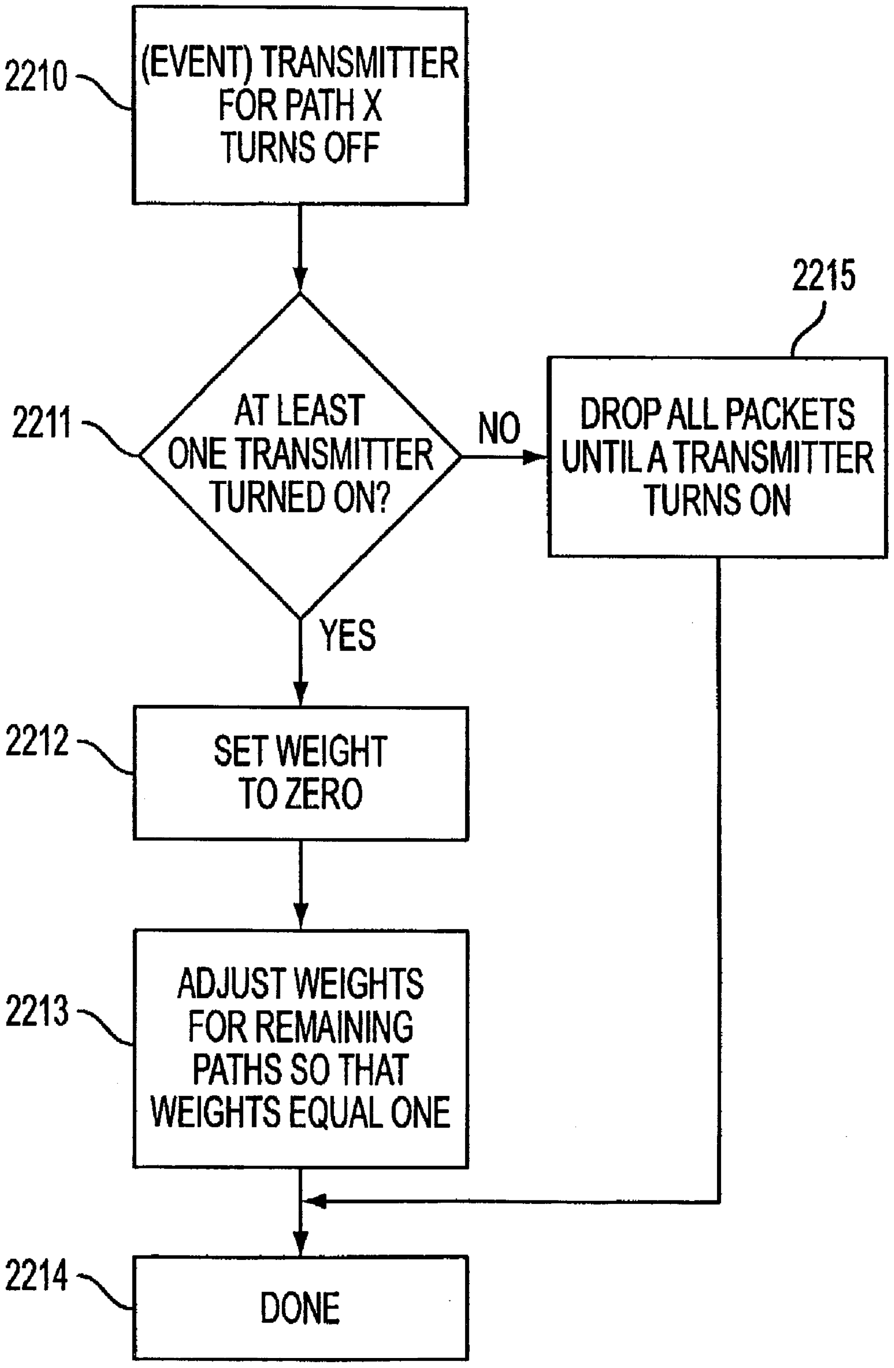

2210
(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2211
AT LEAST ONE TRANSMITTER TURNED ON?
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
2212
SET WEIGHT TO ZERO
2213
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2214
DONE

FIG. 22B

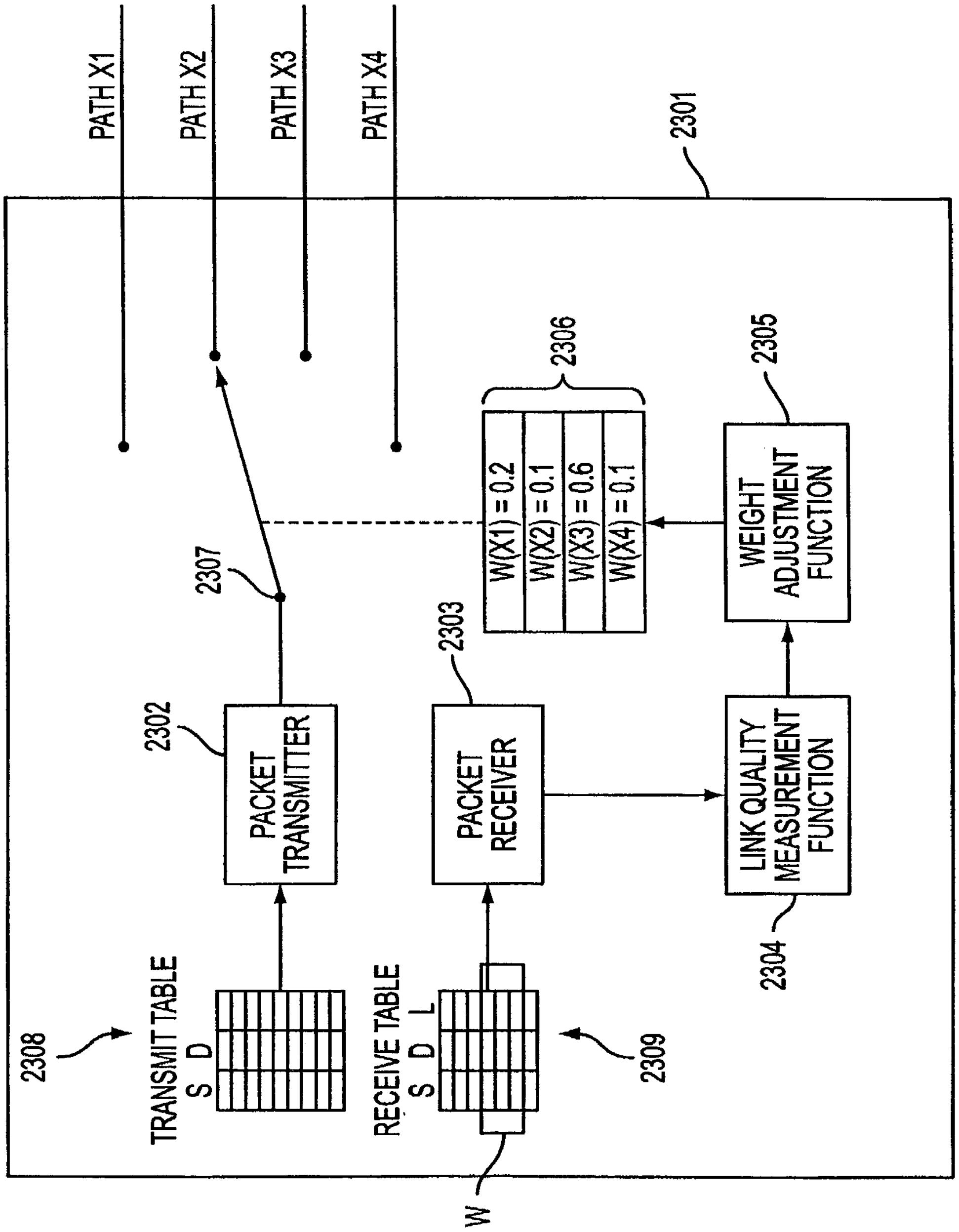

PATH X1
PATH X2
PATH X3
PATH X4
2301
2306
W(X1) = 0.2
W(X2) = 0.1
W(X3) = 0.6
W(X4) = 0.1
2305
WEIGHT ADJUSTMENT FUNCTION
2307
2303
2302
PACKET TRANSMITTER
PACKET RECEIVER
LINK QUALITY MEASUREMENT FUNCTION
2304
TRANSMIT TABLE
S D
2308
RECEIVE TABLE
S D L
2309
W

FIG. 23

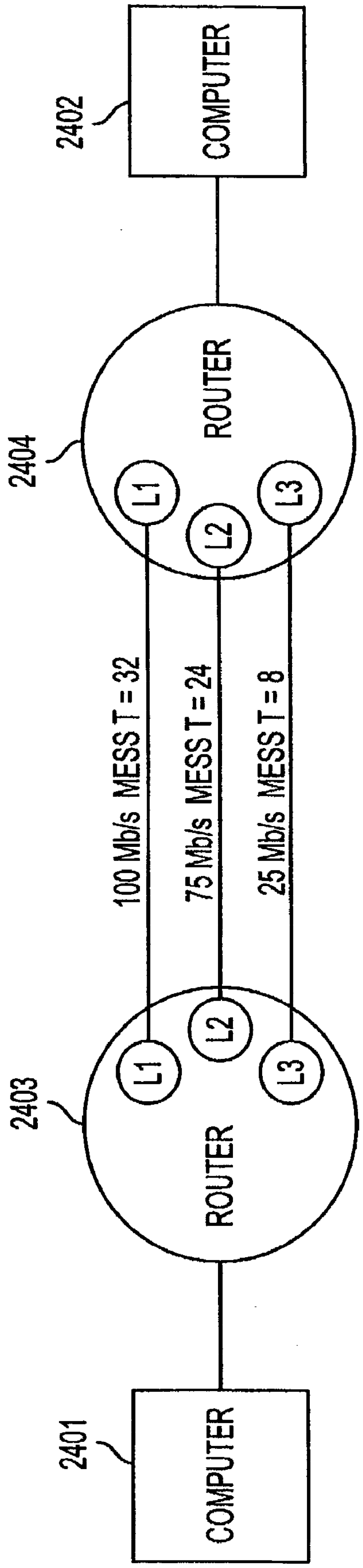

2402
COMPUTER
2404
ROUTER
L1
L2
L3
100 Mb/s MESS T = 32
75 Mb/s MESS T = 24
25 Mb/s MESS T = 8
2403
L1
L2
L3
ROUTER
2401
COMPUTER

FIG. 24

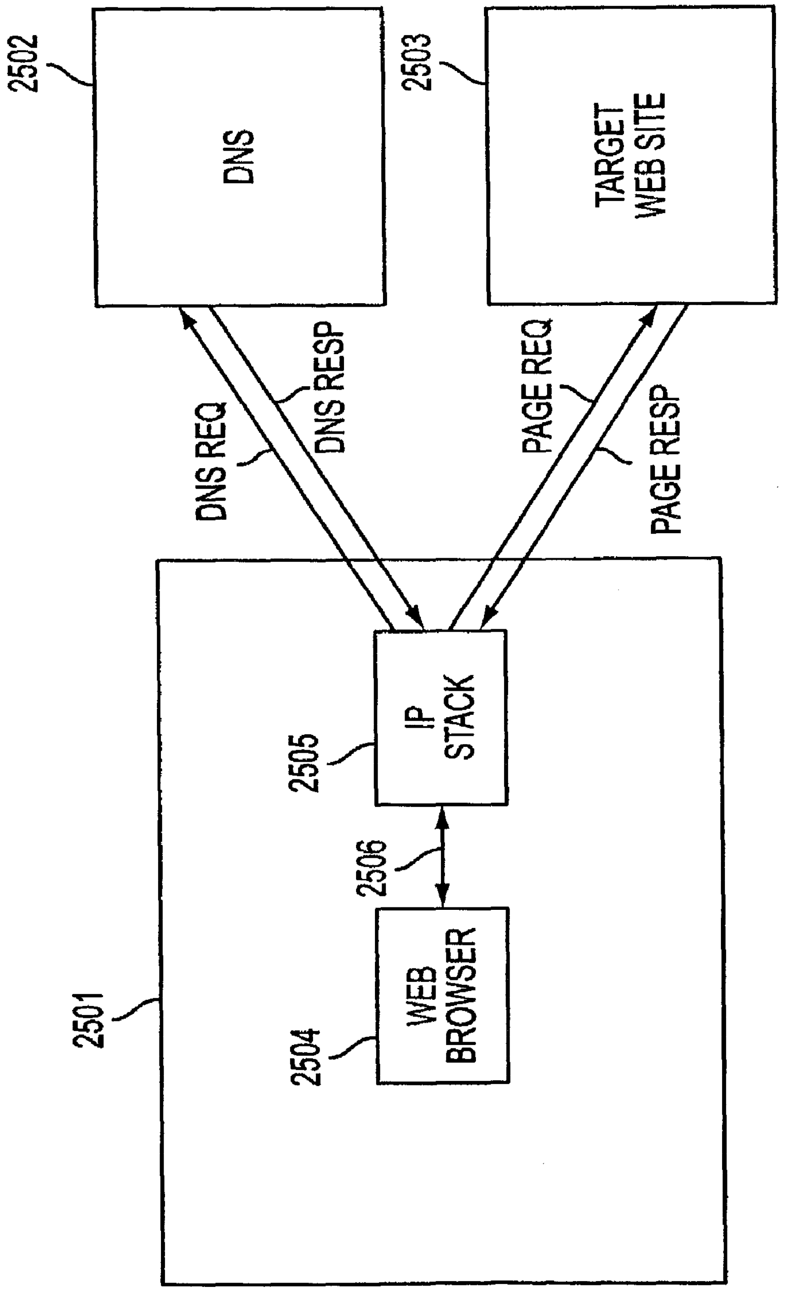


FIG. 25
(PRIOR ART)

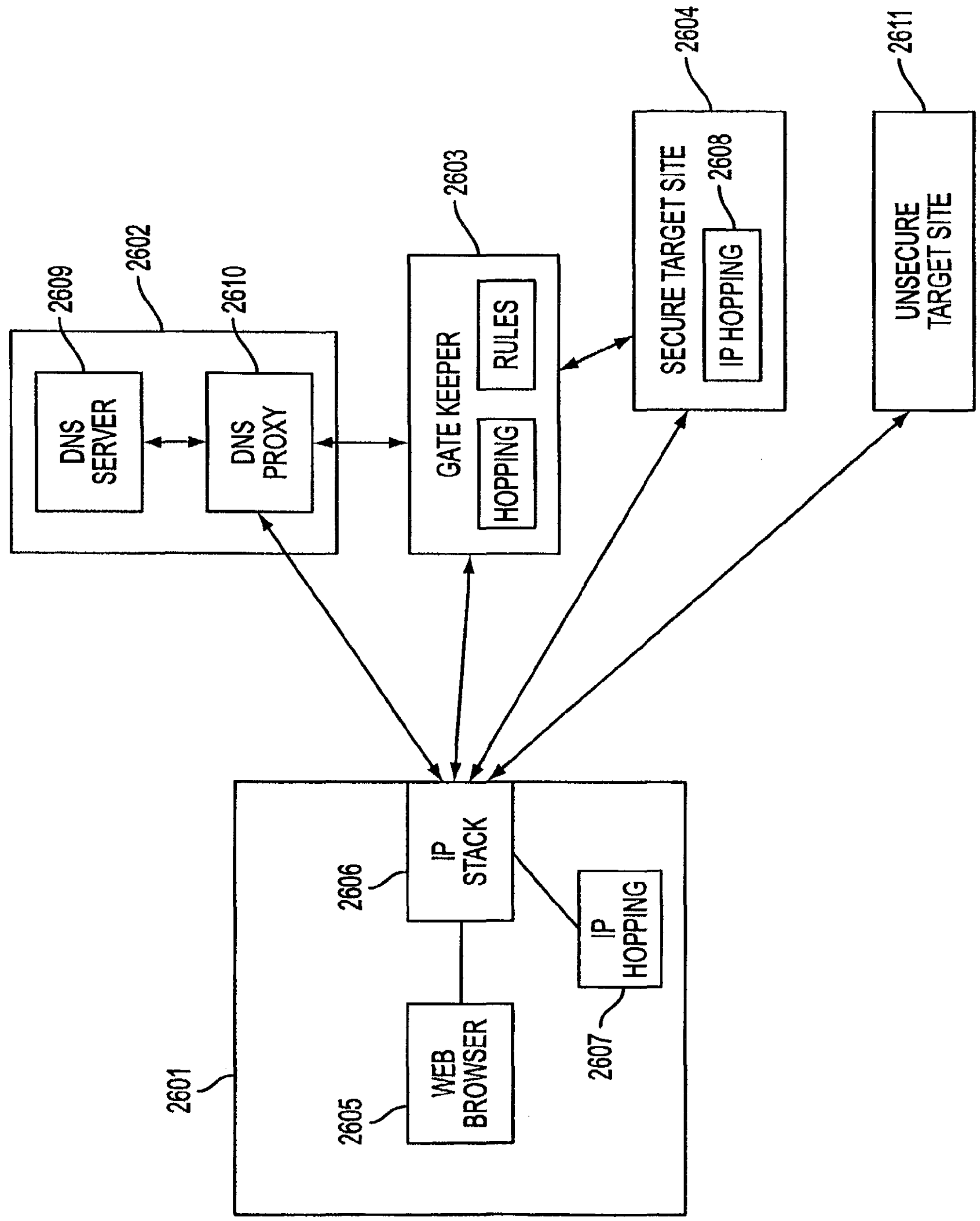

2602
2609
DNS SERVER
2610
DNS PROXY
2603
GATE KEEPER
RULES
HOPPING
2604
SECURE TARGET SITE
2608
IP HOPPING
2611
UNSECURE TARGET SITE
2601
2606
IP STACK
IP HOPPING
2607
2605
WEB BROWSER

FIG. 26

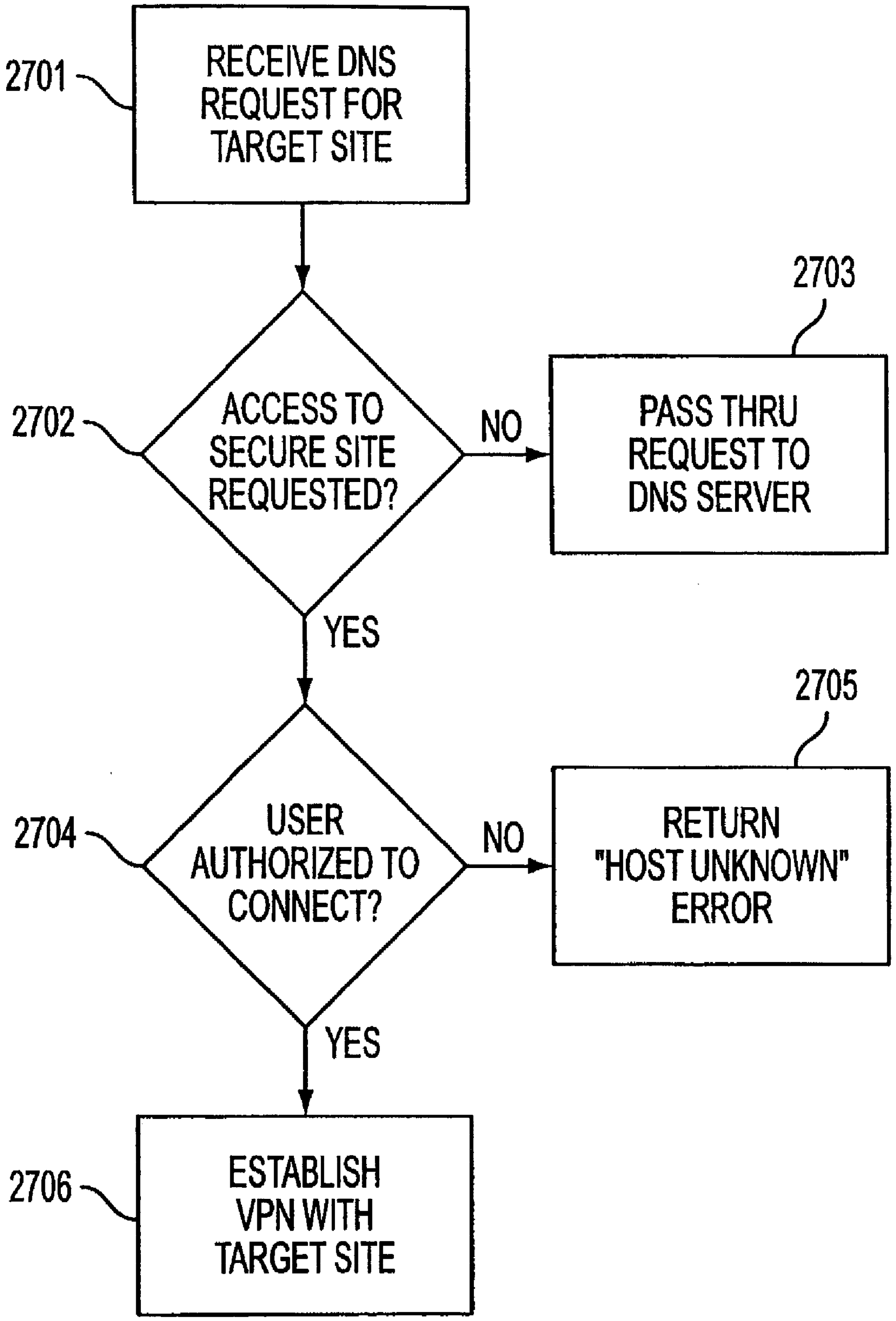

2701
RECEIVE DNS REQUEST FOR TARGET SITE
2702
ACCESS TO SECURE SITE REQUESTED?
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
2704
USER AUTHORIZED TO CONNECT?
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
2706
ESTABLISH VPN WITH TARGET SITE

FIG. 27

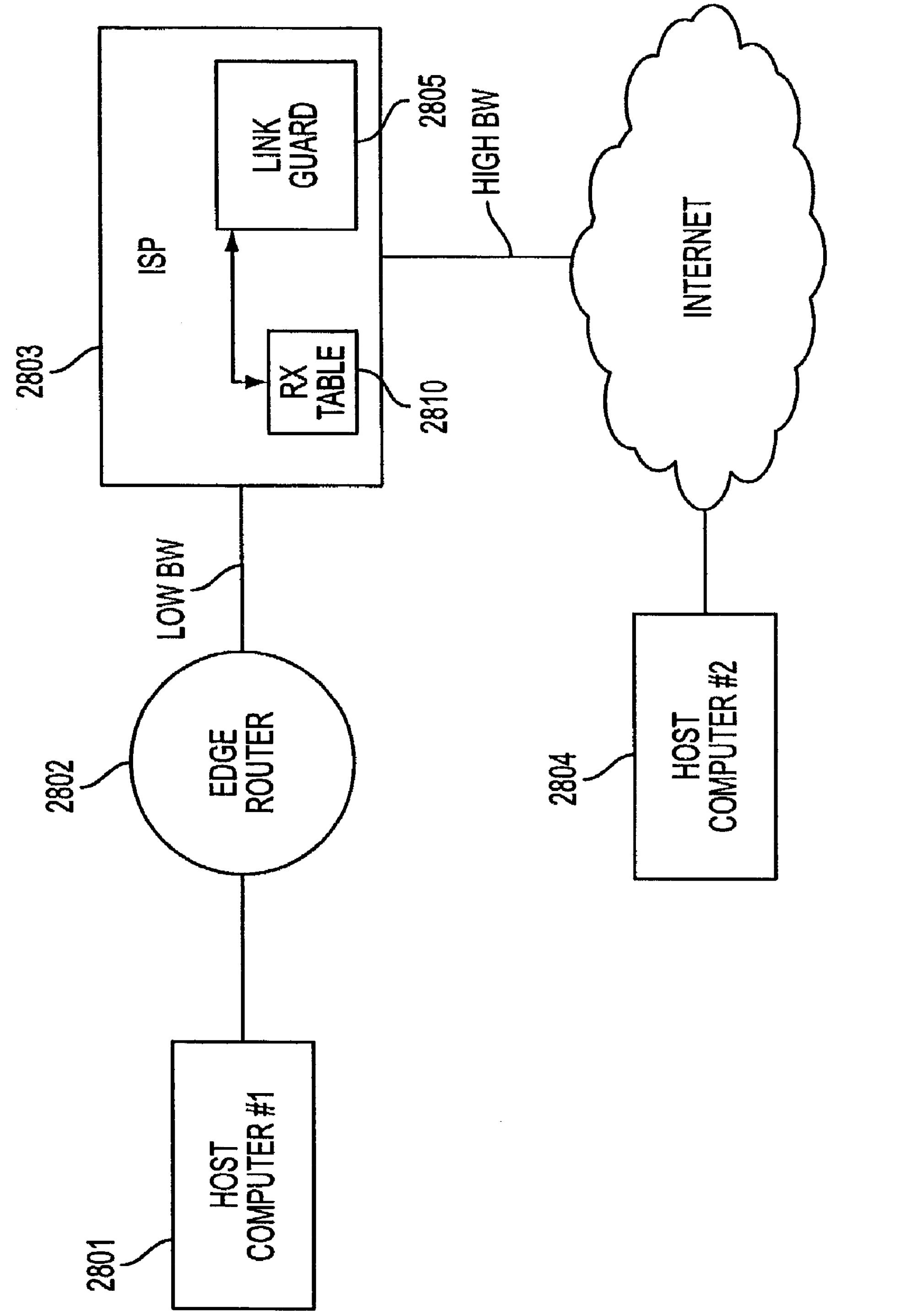
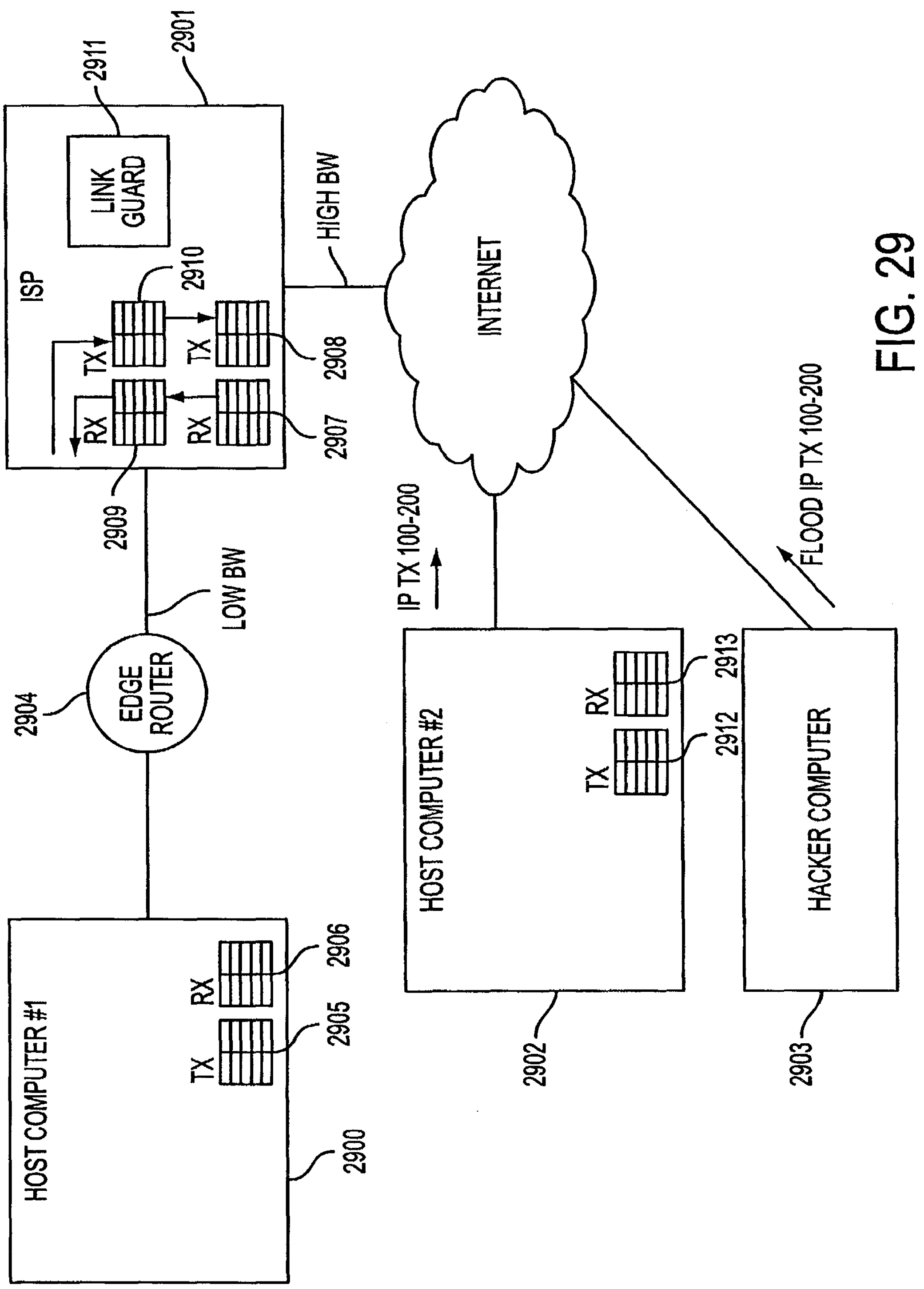

2803
ISP
LINK GUARD
2805
RX TABLE
2810
HIGH BW
INTERNET
LOW BW
2802
EDGE ROUTER
2804
HOST COMPUTER #2
2801
HOST COMPUTER #1

FIG. 28



ISP
LINK GUARD
2911
2901
2910
TX
TX
RX
RX
2908
2907
2909
HIGH BW
INTERNET
LOW BW
EDGE ROUTER
2904
IP TX 100-200
FLOOD IP TX 100-200
HOST COMPUTER #1
TX
RX
2905
2906
2900
HOST COMPUTER #2
TX
RX
2912
2913
2902
HACKER COMPUTER
2903

FIG. 29

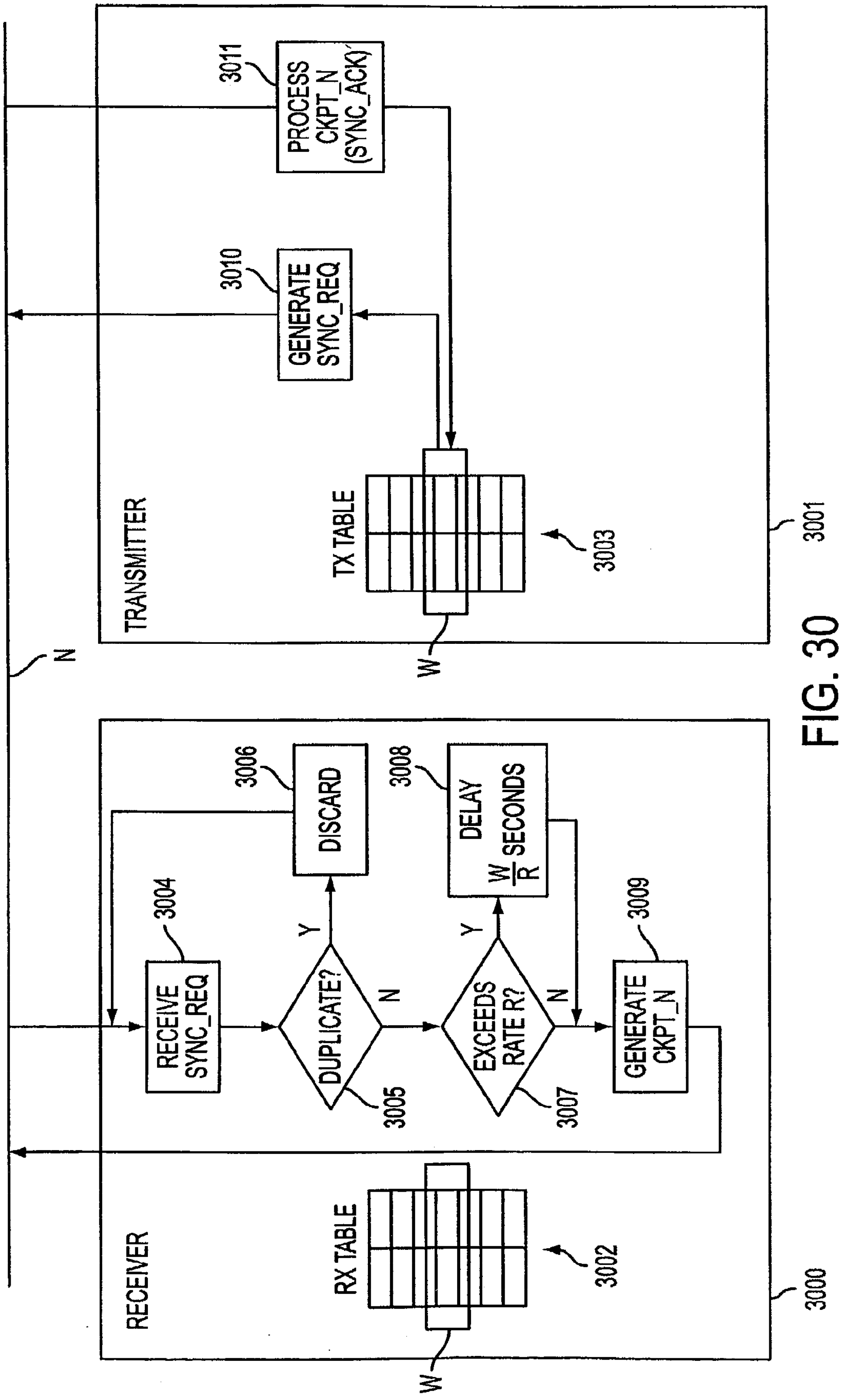

RECEIVER
3000
RX TABLE
3002
W
RECEIVE SYNC_REQ
3004
DUPLICATE?
3005
Y
N
DISCARD
3006
EXCEEDS RATE R?
3007
Y
N
DELAY $\frac{W}{R}$ SECONDS
3008
GENERATE CKPT_N
3009
N
TRANSMITTER
3001
TX TABLE
3003
W
GENERATE SYNC_REQ
3010
PROCESS CKPT_N (SYNC_ACK)
3011

FIG. 30

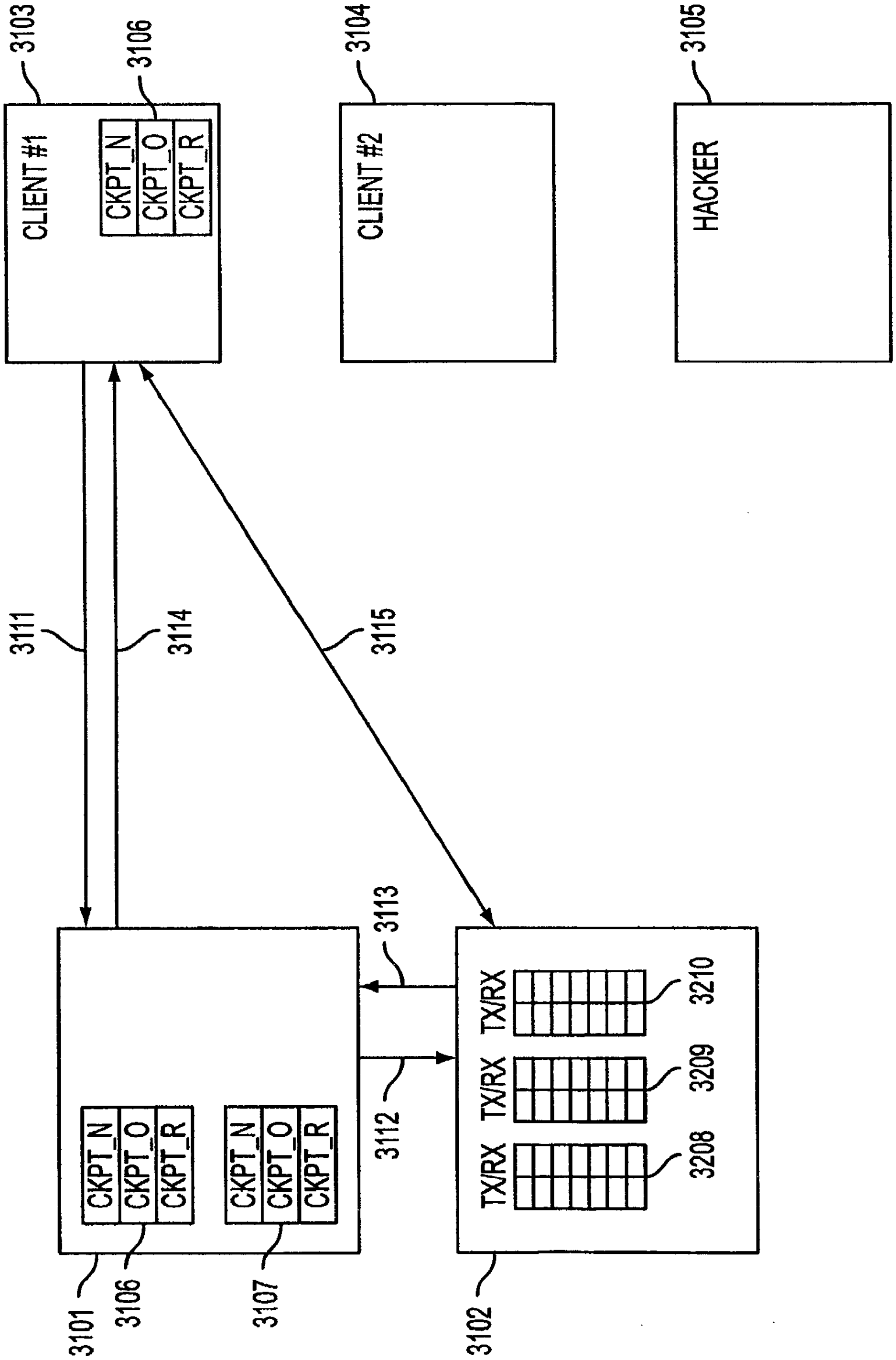

3103
3106
CLIENT #1
CKPT_N
CKPT_O
CKPT_R
3104
CLIENT #2
3105
HACKER
3111
3114
3115
3113
3112
TX/RX
TX/RX
TX/RX
3210
3209
3208
CKPT_N
CKPT_O
CKPT_R
CKPT_N
CKPT_O
CKPT_R
3101
3106
3107
3102

FIG. 31

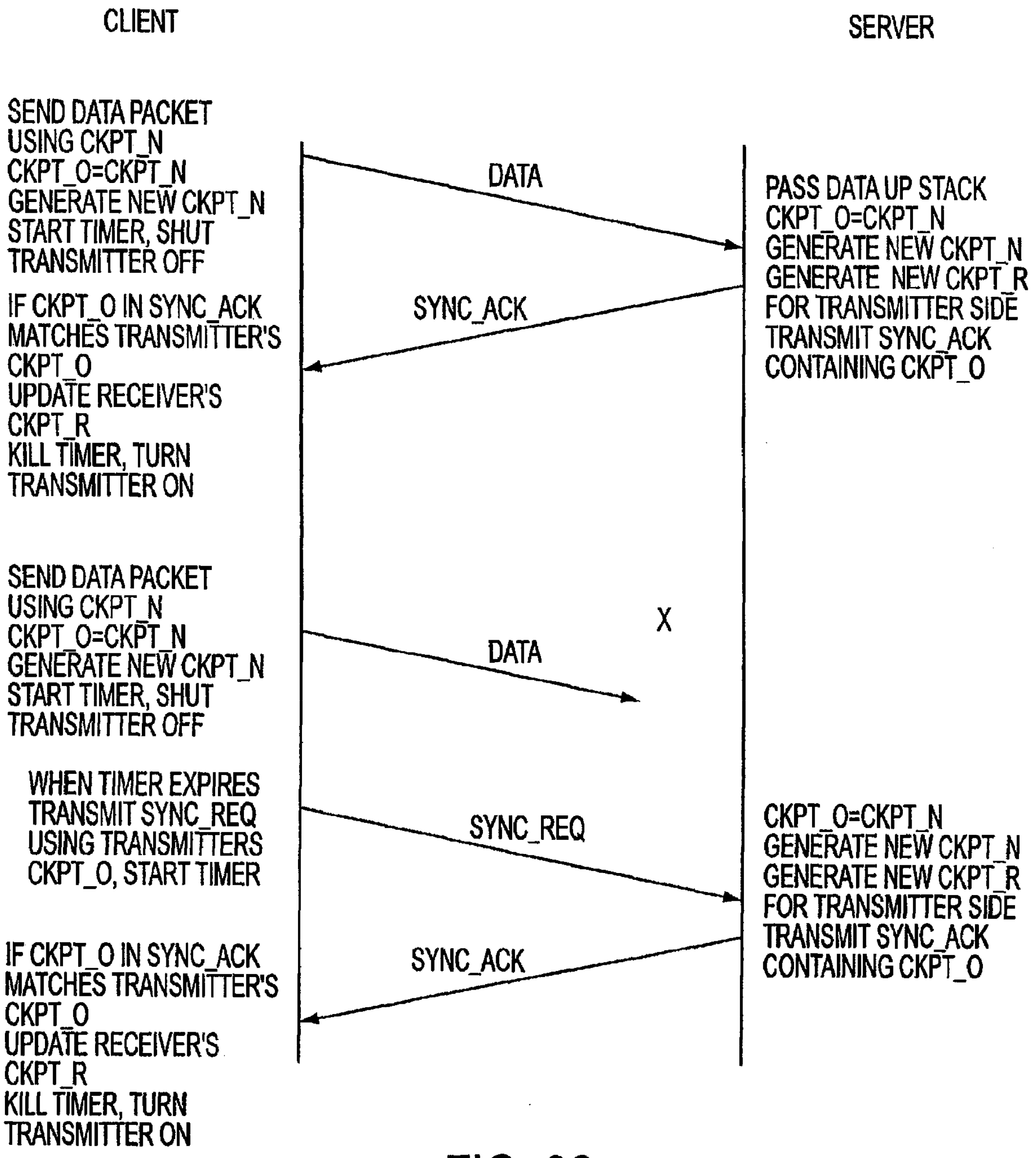

CLIENT
SERVER
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
DATA
PASS DATA UP STACK
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
X
DATA
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
CKPT_O, START TIMER
SYNC_REQ
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a divisional application of 09/504,783 (filed Feb. 15, 2000), now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part of previously filed U.S. application Ser. No. 09/429,643 (filed Oct. 29, 1999) now U.S. Pat. No. 7,010,604. The subject matter of the '643 application, which is bodily incorporated herein, derives from provisional U.S. application No. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999).

GOVERNMENT CONTRACT RIGHTS

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. **1**. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked

using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12A** shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12B** shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. 20 shows a system employing distributed transmission paths.

FIG. 21 shows a plurality of link transmission tables that can be used to route packets in the system of FIG. 20.

FIG. 22A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. 22B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP_C. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers 122-127 intervening between the originating 100 and destination 110 TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. 3*a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1**-**9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. 3*b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. 3*b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. 3*a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. 3*a*. The remaining process is as shown in, and discussed with reference to, FIG. 3*a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IP_T, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP_C is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. 4, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122**-**127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

1. SCALABILITY ENHANCEMENTS

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client 1001 can establish three simultaneous sessions with each of three TARP routers provided by different ISPs 1011, 1012, 1013. As an example, the client 1001 can use three different telephone lines 1021, 1022, 1023 to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. FURTHER EXTENSIONS

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. 11, for example, a first Ethernet frame 1150 comprises a frame header 1101 and two embedded IP packets IP1 and IP2, while a second Ethernet frame 1160 comprises a different frame header 1104 and a single IP packet IP3. Each frame header generally includes a source hardware address 1101A and a destination hardware address 1101B; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. **12A** shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to thc exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. **12A**, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardwarc component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN-which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an

address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. **12A** and **12B** in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208**X that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window W3 that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window W3 slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window W3 will be accepted; those falling outside of window W3 will be rejected as invalid. The length of window W3 can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221**X, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209**X. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window W1 maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12**A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101**A and **1101**B in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12**B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N-1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. **14**).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. **15**. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers X_1, X_2, X_3 . . . X_k starting with seed X_0 using a recurrence

$$X_i=(a\,X_{i-1}+b) \bmod c \qquad (1)$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1)) \bmod c \qquad (2)$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c \qquad (3)$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c=((a^i \bmod((a-1)frxc)\ (X_0(a-1)+b)-b)/(a-1)) \bmod c \qquad (4)$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b)$ mod c, b as b mod c and compute a^i mod ((a−1)c)(this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the j^{th} checkpoint, as X_0 and n as i, a node can store a^n mod((a−1)c) once per LCR and set

$$X_{j+1}{}^w=X_{n(j+1)}=((a^n \bmod((a+1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \qquad (5)$$

to generate the random number for the $j+1^{th}$ synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31\,X_{i-1}+4) \bmod 15. \qquad (6)$$

If one sets X_0=1, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, c*(a−1)=15*30=450 and a^n mod ((a−1)c)=31^3mod(15*30)=29791mod(450)=91. Equation (5) becomes:

$$(91(X_i30+4)-4)/30) \bmod 15 \qquad (7)$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91\ (X_i30 + 4) - 4$	$((91\ (X_i30 + 4) - 4) / 30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Alorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^n-1). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. **16**). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses (1≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧1). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. **17** depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. **18** when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. **19** shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE –OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. CONTINUATION-IN-PART IMPROVEMENTS

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. **22**A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. **22**A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22**A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. **22**B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to asses the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P'=\alpha\times MIN+(1-\alpha)\times P \qquad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P'=\beta\times S+(1-\beta)\times P \qquad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link **1** would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. **25**. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC **2535**).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **2704**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **2703** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **2701**, a DNS look-up request is received for a target host. In step **2702**, a check is made to determine whether access to a secure host was requested. If not, then in step **2703** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **2702**, if access to a secure host was requested, then in step **2704** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **2705**). If the user has sufficient security privileges, then in step **2706** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid.

According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**]. According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until $M \times N \times W/R$ seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC_REQ every T1 seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of $M \times N \times W/R$ seconds after the last SYNC_REQ has been received and accepted, $2 \times M \times N \times W/R$ seconds after next to the last SYNC_REQ has been received and accepted, $C \times M \times N \times W/R$ seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer T1 noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.
2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and and other information (i.e user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side

CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPT_O associated with the timer, it starts timer T1 noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.
5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

We claim:

1. A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:

(i) determining whether the intercepted DNS request corresponds to a secure server;

(ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer, and

(iii) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

2. The data processing device of claim **1**, wherein step (iii) comprises the steps of:

(a) determining whether the client is authorized to access the secure server; and

(b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.

3. The data processing device of claim **2**, wherein step (iii) further comprises the step of:

(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

4. The data processing device of claim **3**, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

5. The data processing device of claim **1**, wherein automatically initiating the encrypted channel between the client and the secure sewer comprises establishing an IF address hopping scheme between the client and the secure server.

6. The data processing device of claim **1**, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

7. A computer readable medium storing a domain name server (DNS) proxy module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

(i) intercepting a DNS request sent by a client;

(ii) determining whether the intercepted DNS request corresponds to a secure server;

(iii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and

(iv) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

8. The computer readable medium of claim **7**, wherein step (iv) comprises the steps of

(a) determining whether the client is authorized to access the secure server, and

(b) when the client is authorized to access the secure server, sending a request to the secure sewer to establish an encrypted channel between the secure sewer and the client.

9. The computer readable medium of claim **8**, wherein step (iv) further comprises the step of:

(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

10. The computer readable medium of claim **9**, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

11. The computer readable medium of claim **7**, wherein automatically initiating the encrypted channel between the client and the secure sewer comprises establishing an IP address hopping scheme between the client and the secure server.

12. The computer readable medium of claim **7**, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

13. A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

(i) determining whether a DNS request sent by a client corresponds to a secure server;

(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and

(iii) when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel between the client and the secure server.

14. The computer readable medium of claim **13**, wherein step (iii) comprises the steps of

(a) determining whether the client is authorized to access the secure server; and

(b) when the client is authorized to access the secure server, sending a request to the secure server to establish a secure channel between the secure server and the client.

15. The computer readable medium of claim **14**, wherein step (iii) further comprises the step of:

(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

16. The computer readable medium of claim **15**, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

* * * * *

TAB 9

U 7290575

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

April 12, 2011

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *7,921,211*
ISSUE DATE: *April 05, 2011*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

W. Montgomery

W. MONTGOMERY
Certifying Officer

Plaintiffs' VirnetX Exhibit
VirnetX, Inc. v. Apple, Inc.
PX010
C.A. 6:10-cv-0417

US007921211B2

(12) **United States Patent**
Larson et al.

(10) **Patent No.: US 7,921,211 B2**
(45) **Date of Patent: *Apr. 5, 2011**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES**

(75) Inventors: **Victor Larson**, Fairfax, VA (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Edmund Colby Munger**, Crownsville, MD (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **VirnetX, Inc.**, Scotts Valley, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 701 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/840,560**

(22) Filed: **Aug. 17, 2007**

(65) **Prior Publication Data**

US 2008/0040792 A1 Feb. 14, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/714,849, filed on Nov. 18, 2003, now Pat. No. 7,418,504, which is a continuation of application No. 09/558,210, filed on Apr. 26, 2000, now abandoned, which is a continuation-in-part of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/226**

(58) **Field of Classification Search** 709/226, 709/221; 726/15
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,895,502	A	7/1959	Roper et al.
5,303,302	A	4/1994	Burrows
5,311,593	A	5/1994	Carmi

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0838930	4/1988

(Continued)

OTHER PUBLICATIONS

Baumgartner et al, "Differentiated Services: A New Approach for Quality of Service in the Internet," International Conference on High Performance Networking, 255-273 (1998).

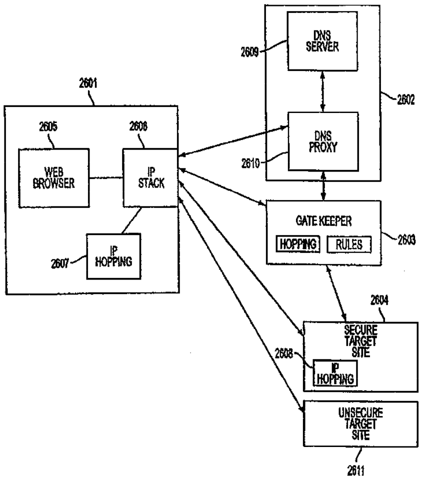
(Continued)

Primary Examiner — Krisna Lim
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A secure domain name service for a computer network is disclosed that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. The portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

60 Claims, 40 Drawing Sheets



U.S. PATENT DOCUMENTS

5,384,848	A	1/1995	Kikuchi	
5,511,122	A	4/1996	Atkinson	
5,629,984	A	5/1997	McManis	
5,764,906	A	6/1998	Edelstein et al.	
5,771,239	A	6/1998	Moroney et al.	
5,805,803	A	9/1998	Birrell et al.	
5,822,434	A	10/1998	Caronni et al.	
5,864,666	A *	1/1999	Shrader	726/15
5,870,610	A	2/1999	Beyda et al.	
5,898,830	A	4/1999	Wesinger, Jr. et al.	
5,950,195	A	9/1999	Stockwell et al.	
6,052,788	A	4/2000	Wesinger et al.	
6,055,574	A	4/2000	Smorodinsky et al.	
6,061,346	A	5/2000	Nordman	
6,079,020	A	6/2000	Liu	
6,081,900	A *	6/2000	Subramaniam et al.	726/19
6,101,182	A	8/2000	Sistanizadeh et al.	
6,119,171	A	9/2000	Alkhatib	
6,173,399	B1	1/2001	Gilbrech	
6,199,112	B1	3/2001	Wilson	
6,202,081	B1	3/2001	Naudus	
6,223,287	B1	4/2001	Douglas et al.	
6,226,748	B1	5/2001	Bots et al.	
6,226,751	B1	5/2001	Arrow et al.	
6,246,670	B1	6/2001	Karlsson et al.	
6,262,987	B1	7/2001	Mogul	
6,298,341	B1	10/2001	Mann et al.	
6,314,463	B1	11/2001	Abbott et al.	
6,333,272	B1	12/2001	McMillin et al.	
6,338,082	B1	1/2002	Schneider	
6,502,135	B1	12/2002	Munger et al.	
6,557,037	B1	4/2003	Provino	
6,687,746	B1	2/2004	Shuster et al.	
6,701,437	B1	3/2004	Hoke et al.	
6,752,166	B2	6/2004	Lull et al.	
6,757,740	B1	6/2004	Parkh et al.	
6,937,597	B1	8/2005	Rosenberg et al.	
7,039,713	B1	5/2006	Van Gunter et al.	
7,072,964	B1	7/2006	Whittle et al.	
7,167,904	B1	1/2007	Devarajan et al.	
7,188,175	B1	3/2007	McKeeth	
7,353,841	B2	4/2008	Kono et al.	
7,461,334	B1	12/2008	Lu et al.	
7,490,151	B2	2/2009	Munger et al.	
7,493,403	B2	2/2009	Shull et al.	
2001/0049741	A1	12/2001	Skene et al.	
2004/0199493	A1	10/2004	Ruiz et al.	
2004/0199520	A1	10/2004	Ruiz et al.	
2004/0199608	A1	10/2004	Rechterman et al.	
2004/0199620	A1	10/2004	Ruiz et al.	
2007/0208869	A1	9/2007	Adelman et al.	
2007/0214284	A1	9/2007	King et al.	
2007/0266141	A1	11/2007	Norton	
2008/0235507	A1	9/2008	Ishikawa et al.	

FOREIGN PATENT DOCUMENTS

EP	0814589	12/1997
GB	2317792	4/1998
GB	2334181	8/1999
GB	2340702	2/2000
JP	62-214744	9/1987
JP	04-363941	12/1992
JP	09-018492	1/1997
JP	10-070531	3/1998
WO	WO98/27783	6/1998
WO	WO99/11019	3/1999
WO	WO 00/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01/16766	3/2001

OTHER PUBLICATIONS

Chapman et al., "Domain Name System (DNS)," 278-296 (1995).

Davila et al., "Implementation of Virtual Private Networks at the Transport Layer," M. Mambo, Y. Zheng (Eds), Information Security (Second International) Workshop, ISW' 99. Lecture Notes in Computer Science (LNCS), vol. 1729; 85-102 (1999).

De Raadt et al., "Cryptography in OpenBSD," 10 pages (1999).

Eastlake, "Domain Name System Security Extensions," Internet Citation, Retrieved from the Internet: URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-dnssec-secext2-05.txt (1998).

Gunter et al., "An Architecture for Managing QoS-Enabled VRNs Over the Internet," Proceedings 24th Conference on Local Computer Networks. LCN' 99 IEEE Comput. Soc Los Alamitos, CA, pp. 122-131 (1999).

Shimizu, "Special Feature: Mastering the Internet with Windows 2000", Internet Magazine, 63:296-307 (2000).

Stallings, "Cryptography and Network Security," Principals and Practice, 2nd Edition, pp. 399-440 (1999).

Takata, "U.S. Vendors Take Serious Action to Act Against Crackers—A Tracking Tool and a Highly Safe DNS Software are Released", Nikkei Communications, 257:87(1997).

Wells, Email (Lancasterb1be@mail.msn.com), Subject: "Security Icon," (1998).

Fasbender, A., et al., Variable and Scalable Security: Protection of Location Information in Mobile IP, IEEE VTS, 46th, 1996, 5 pp.

DNS-related correspondence dated Sep. 7, 1993 to Sep. 20, 1993. (Pre KX, KX Records).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 2, 1996). (RFC 2543 Internet Draft 1).

Aventail Corp., "AutoSOCKS v. 2.1 Datasheet," available at http://www.archive.org/web/19970212013409/www.aventail.com/prod/autosk2ds.html (1997). (AutoSOCKS, Aventail).

Aventail Corp., "Socks Version 5," Aventail Whitepaper, available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/soc kswp.html (1997). (Socks, Aventail).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Mar. 27, 1997). (RFC 2543 Internet Draft 2).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 31, 1997). (RFC 2543 Internet Draft 3).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 11, 1997). (RFC 2543 Internet Draft 4).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (May 14, 1998). (RFC 2543 Internet Draft 5).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jun. 17, 1998). (RFC 2543 Internet Draft 6).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 16, 1998). (RFC 2543 Internet Draft 7).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Aug. 7, 1998). (RFC 2543 Internet Draft 8).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Sep. 18, 1998). (RFC 2543 Internet Draft 9).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 12, 1998). (RFC 2543 Internet Draft 10).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 15, 1998). (RFC 2543 Internet Draft 11).

Aventail Corp., "Aventail Connect 3.1/2.6Administrator's Guide," (1999). (Aventail Administrator 3.1, Aventail).

Aventail Corp., "Aventail Connect 3.1/2.6 User's Guide," (1999). (Aventail User 3.1, Aventail).

Aventail Corp., "Aventail ExtraWeb Server v3.2 Administrator's Guide," (1999). (Aventail ExtraWeb 3.2, Aventail).

Check Point Software Technologies Ltd. (1999) (Check Point, Checkpoint FW).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jan. 15, 1999). (RFC 2543 Internet Draft 12).

Goncalves, et al. *Check Point FireWall—1 Administration Guide*, McGraw-Hill Companies (2000). (Goncalves, Checkpoint FW).

Assured Digital Products. (Assured Digital).

F-Secure, *F-Secure Evaluation Kit* (May 1999) (FSECURE 00000003) (Evaluation Kit 3).

F-Secure, *F-Secure Evaluation Kit* (Sep. 1998) (FSECURE 00000009) (Evaluation Kit 9).

IRE, Inc., *SafeNet/Soft-PK Version 4* (Mar. 28, 2000) (Soft-PK Version 4).

IRE/SafeNet Inc., *VPN Technologies Overview* (Mar. 28, 2000) (Safenet VPN Overview).

IRE, Inc., *SafeNet/VPN Policy Manager Quick Start Guide Version 1* (1999) (SafeNet VPN Policy Manager).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).

U.S. Appl. No. 60/134,547, filed May 17, 1999, Victor Sheymov.

U.S. Appl. No. 60/151,563, filed Aug. 31, 1999, Bryan Whittles.

U.S. Appl. No. 09/399,753, filed Sep. 22, 1998, Graig Miller et al.

Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009, *VirnetX Inc. and Science Applications International Corp.* v. *Microsoft Corporation.*

Appendix A of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

Concordance Table for the References Cited in Tables on pp. 6-15, 71-80 and 116-124 of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

1. P. Mockapetris, "DNS Encoding of Network Names and Other Types," Network Working Group, RFC 1101 (Apr. 1989) (RFC1101, DNS SRV).

R. Atkinson, "An Internetwork Authentication Architecture," Naval Research Laboratory, Center for High Assurance Computing Systems (Aug. 5, 1993). (Atkinson NRL, KX Records).

Henning Schulzrinne, *Personal Mobility for Multimedia Services In The Internet*, Proceedings of the Interactive Distributed Multimedia Systems and Services European Workshop at 143 (1996). (Schulzrinne 96).

Microsoft Corp., *Microsoft Virtual Private Networking: Using Point-to-Point Tunneling Protocol for Low-Cost, Secure, Remote Access Across the Internet* (1996) (printed from 1998 PDC DVD-ROM). (Point to Point, Microsoft Prior Art VPN Technology).

"Safe Surfing: How to Build a Secure World Wide Web Connection," IBM Technical Support Organization, (Mar. 1996). (Safe Surfing, Website Art).

Goldschlag, et al., "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK (May 1996). (Goldschlag II, Onion Routing).

"IPSec Minutes From Montreal", IPSEC Working Group Meeting Notes, http://www.sandleman.ca/ipsec/1996/08/msg00018.html (Jun. 1996). (IPSec Minutes, FreeS/WAN).

J. M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose, California, Jul. 1996. (Galvin, DNSSEC).

J. Gilmore, et al. "Re: Key Management, anyone? (DNS Keying)," IPSec Working Group Mailing List Archives (Aug. 1996). (Gilmore DNS, FreeS/WAN).

H. Orman, et al. "Re: 'Re: DNS? was Re: Key Management, anyone?" IETF IPSec Working Group Mailing List Archive (Aug. 1996-Sep. 1996). (Orman DNS, FreeS/WAN).

Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2052 (Oct. 1996). (RFC 2052, DNS SRV).

Freier, et al. "The SSL Protocol Version 3.0," Transport Layer Security Working Group (Nov. 18, 1996). (SSL, Underlying Security Technology).

M.G. Reed, et al. "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference, San Diego, CA, Dec. 9-13, 1996. (Reed, Onion Routing).

Kenneth F. Alden & Edward P. Wobber, *The AltaVista Tunnel: Using the Internet to Extend Corporate Networks*, Digital Technical Journal (1997) (Alden, AltaVista.

Automotive Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).

Automotive Industry Action Group, "ANX Release 1 Draft Document Publication," AIAG Publications (1997). (AIAG Release, ANX).

Aventail Corp. "Aventail VPN Data Sheet," available at http://www.archive.org/web/19970212013043/www.aventail.com/prod/vpndata.html (1997).(Data Sheet, Aventail).

Aventail Corp., "Directed VPN Vs. Tunnel," available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/directvpn.html (1997). (Directed VPN, Aventail).

Aventail Corp., "Managing Corporate Access to the Internet," Aventail AutoSOCKS White Paper *available at* http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/ipmwp.html (1997). (Corporate Access, Aventail).

Aventail Corp., "VPN Server V2.0 Administration Guide," (1997). (VPN, Aventail).

Goldschlag, et al. "*Privacy on the Internet*," Naval Research Laboratory, Center for High Assurance Computer Systems (1997). (Goldschlag I, Onion Routing).

Microsoft Corp., *Installing Configuring and Using PPTP with Microsoft Clients and Servers* (1997). (Using PPTP, Microsoft Prior Art VPN Technology).

Microsoft Corp., *IP Security for Microsoft Windows NT Server 5.0* (1997) (printed from 1998 PDC DVD-ROM). (IP Security, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Microsoft Windows NT Active Directory: An Introduction to the Next Generation Directory Services* (1997) (printed from 1998 PDC DVD-ROM). (Directory, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Routing and Remote Access Service for Windows NT Server NewOpportunities Today and Looking Ahead* (1997) (printed from 1998 PDC DVD-ROM).(Routing, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Understanding Point-to-Point Tunneling Protocol PPTP* (1997) (printed from 1998 PDC DVD-ROM). (Understanding PPTP, Microsoft Prior Art VPN Technology).

J. Mark Smith et.al., *Protecting a Private Network: The AltaVista Firewall*, Digital Technical Journal (1997). (Smith, AltaVista).

Naganand Doraswamy *Implementation of Virtual Private Networks (VPNs) with IPSecurity*, <draft-ietf-ipsec-vpn-00.txt> (Mar. 12, 1997). (Doraswamy).

Aventail Corp., "Aventail, and Cybersafe to Provide Secure Authentication For Internet and Intranet Communication," Press Release, Apr. 3, 1997. (Secure Authentication, Aventail).

D. Wagner, et al. "Analysis of the SSL 3.0 Protocol," (Apr. 15, 1997). (Analysis, Underlying Security Technologies).

Automotive Industry Action Group, "ANXO Certification Authority Service and Directory Service Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Defintion, ANX).

Automotive Industry Action Group, "ANXO Certification Process and ANX Registration Process Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Certification, ANX).

Aventail Corp., "Aventail Announces the First VPN Solution to Assure Interoperability Across Emerging Security Protocols," Jun. 2, 1997. (First VPN, Aventail).

Syverson, et al. "Private Web Browsing," Naval Research Laboratory, Center for High 8 Assurance Computer Systems (Jun. 2, 1997). (Syverson, Onion Routing).

Bellcore, "Metrics, Criteria, and Measurement Technique Requirements for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (Jun. 16, 1997). (AIAG Requirements, ANX).

R. Atkinson, "Key Exchange Delegation Record for the DNS," Network Working Group, RFC 2230 (Nov. 1997). (RFC 2230, KX Records).

1998 Microsoft Professional Developers Conference DVD ("1998 PDC DVD-ROM") (including screenshots captured therefrom and produced as MSFTVX 00018827-00018832). (Conference, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Virtual Private Networking An Overview* (1998) (printed from 1998 PDC DVD-ROM) (Overview, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Windows NT 5.0 Beta Has Public Premiere at Seattle Mini-Camp Seminar attendees get first look at the performance and capabilities of Windows NT 5.0* (1998) (available at hap //www.microsoft.com/presspass/features/1998/10-19nt5.mspxpftrue).(NT Beta, Microsoft Prior Art VPN Technology).

"What ports does SSL use" available at stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html (1998). (Ports, DNS SRV).

Aventail Corp., "Aventail VPN V2.6 Includes Support for More Than Ten Authentication Methods Making Extranet VPN Development Secure and Simple," Press Release, Jan. 19, 1998. (VPN V2.6, Aventail).

R. G. Moskowitz, "Network Address Translation Issues with IPsec," Internet Draft, Internet Engineering Task Force, Feb. 6, 1998. (Moskowitz).

H. Schulzrinne, et al, "Internet Telephony Gateway Location," Proceedings of IEEE INfocom '98, The Conference on Computer Communications, vol. 2 (Mar. 29-Apr. 2, 1998). (Gateway, Schulzrinne).

C. Huitema, 45 al. "Simple Gateway Control Protocol," Version 1.0 (May 5, 1998). (SGCP).

DISA "Secret Internet Protocol Router Network," SIPRNET Program Management Office (D3113) DISN Networks, DISN Transmission Services (May 8, 1998). (DISA, SIPRNET).

D. McDonald, et al. "PF_KEY Key Management API, Version 2," Network Working Group, RFC 2367 (Jul. 1998). (RFC 2367).

Microsoft Corp., *Company Focuses on Quality and Customer Feedback* (Aug. 18, 1998). (Focus, Microsoft Prior Art VPN Technology).

Atkinson, et al. "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998). (RFC 2401, Underlying Security Technologies).

Donald Eastlake, *Domain Name System Security Extensions*, IETF DNS Security Working Group (Dec. 1998). (DNSSEC-7).

Kaufman et al, "Implementing IPsec," (Copyright 1999). (Implementing IPSEC, VPN References).

Network Solutions, Inc. "Enabling SSL," NSI Registry (1999). (Enabling SSL, Underlying Security Technologies).

C. Scott, et al. *Virtual Private Networks*, O'Reilly and Associates, Inc.; 2nd ed. (Jan. 1999). (Scott VPNs).

Goldschlag, et al., "Onion Routing for Anonymous and Private Internet Connections," Naval Research Laboratory, Center for High Assurance Computer Systems (Jan. 28, 1999). (Goldschlag III, Onion Routing).

H. Schulzrinne, "Internet Telephony: architecture and protocols—an IETF perspective," Computer Networks, vol. 31, No. 3 (Feb. 1999). (Telephony, Schulzrinne).

M. Handley, et al. "SIP: Session Initiation Protocol," Network Working Group, RFC 2543 and Internet Drafts (Dec. 1996-Mar. 1999). (Handley, RFC 2543).

FreeS/WAN Project, *Linux FreeS/WAN Compatibility Guide* (Mar. 4, 1999). (FreeS/WAN Compatibility Guide, FreeS/WAN).

Telcordia Technologies, "ANX Release 1 Document Corrections," AIAG (May 11, 1999). (Telcordia, ANX).

Ken Hornstein & Jeffrey Altman, *Distributing Kerberos KDC and Realm Information with DNS* <draft-eitf-cat-krb-dns-locate-oo.txt> (Jun. 21, 1999). (Hornstein, DNS SRV).

Bhattacharya et. al. "An LDAP Schema for Configuration and Administration of IPSec Based Virtual Private Networks (VPNs)", IETF Internet Draft (Oct. 1999). (Bhattcharya LDAP VPN).

B. Patel, et al. "DHCP Configuration of IPSEC Tunnel Mode," IPSEC Working Group, Internet Draft 02 (Oct. 15, 1999). (Patel).

"Building a Microsoft VPN: A Comprehensive Collection of Microsoft Resources," FirstVPN, (Jan. 2000). (FirstVPN Microsoft).

Gulbrandsen, Vixie, & Esibov, *A DNS RR for specifying the location of services* (*DNS SRV*), IETF RFC 2782 (Feb. 2000). (RFC 2782, DNS SRV).

Mitre Organization, "Technical Description," Collaborative Operations in Joint Expeditionary Force Experiment (JEFX) 99 (Feb. 2000). (MITRE, SIPRNET).

H. Schulzrinne, et al. "Application-Layer Mobility Using SIP," Mobile Computing and Communications Review, vol. 4, No. 3. pp. 47-57 (Jul. 2000). (Application, SIP).

Kindred et al, "Dynamic VPN Communities: Implementation and Experience," DARPA Information Survivability Conference and Exposition II (Jun. 2001). (DARPA, VPN Systems).

ANX 101: Basic ANX Service Outline. (Outline, ANX).

ANX 201: Advanced ANX Service. (Advanced, ANX).

Appendix A: Certificate Profile for ANX IPsec Certificates. (Appendix, ANX).

Aventail Corp., "Aventail AutoSOCKS the Client Key to Network Security," Aventail Corporation White Paper. (Network Security, Aventail).

Cindy Moran, "DISN Data Networks: Secret Internet Protocol Router Network (SIPRNet)." (Moran, SIPRNET).

Data Fellows F-Secure VPN+ (F-Secure VPN+).

Interim Operational Systems Doctrine for the Remote Access Security Program (RASP) Secret Dial-In Solution. (RASP, SIPRNET).

Onion Routing, "Investigation of Route Selection Algorithms," available at http://www.onion-router.net/Archives/Route/index.html. (Route Selection, Onion Routing).

Secure Computing, "Bullet-Proofing an Army Net," Washington Technology. (Secure, SIPRNET).

Sparta "Dynamic Virtual Private Network." (Sparta, VPN Systems).

Standard Operation Procedure for Using the 1910 Secure Modems. (Standard, SIPRNET).

Publically available emails relating to FreeS/WAN (MSFTVX00018833-MSFTVX00019206). (FreeS/WAN emails, FreeS/WAN).

Kaufman et al., "Implementing IPsec," (Copyright 1999) (Implementing IPsec).

Network Associates *Gauntlet Firewall For Unix User's Guide Version 5.0* (1999). (Gauntlet User's Guide—Unix, Firewall Products).

Network Associates *Gauntlet Firewall for Windows NT Getting Started Guide Version 5.0* (1999) (Gauntlet Getting Started Guide—NT, Firewall Products).

Network Associates *Gauntlet Firewall for Unix Getting Started Guide Version 5.0* (1999) (Gauntlet Unix Getting Started Guide, Firewall Products).

Network Associates *Release Notes Gauntlet Firewall for Unix 5.0* (Mar. 19, 1999) (Gauntlet Unix Release Notes, Firewall Products).

Network Associates *Gauntlet Firewall For Windows NT Administrator's Guide Version 5.0* (1999) (Gauntlet NT Administrator's Guide, Firewall Products).

Trusted Information Systems, Inc. *Gauntlet Internet Firewall Firewall-to-Firewall Encryption Guide Version 3.1* (1996) (Gauntlet Firewall-to-Firewall, Firewall Products).

Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).

Network Associates *Gauntlet Firewall For UNIX Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).

Dan Sterne *Dynamic Virtual Private Networks* (May 23, 2000) (Sterne DVPN, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks* (*DVPN*) (Dec. 21, 1999) (Kindred DVPN, DVPN).

Dan Sterne et.al. *TIS Dynamic Security Perimeter Research Project Demonstration* (Mar. 9, 1998) (Dynamic Security Perimeter, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks Capability Description* (Jan. 5, 2000) (Kindred DVPN Capability, DVPN) 11. Oct. 7, and 28, 1997 email from Domenic J. Turchi Jr. (SPARTA00001712-1714, 1808-1811) (Turchi DVPN email, DVPN).

James Just & Dan Sterne *Security Quickstart Task Update* (Feb. 5, 1997) (Security Quickstart, DVPN).

Virtual Private Network Demonstration dated Mar. 21, 1998 (SPARTA00001844-54) (DVPN Demonstration, DVPN).

GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration* (*IFD*) *1.1 Plan* (Mar. 10, 1998) (IFD 1.1, DVPN).

Microsoft Corp. Windows NT Server Product Documentation: Administration Guide—Connection Point Services, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cpsops.mspx (Connection Point Services) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows. Accordingly, upon information and belief, this reference is prior art to the patents-insuit.).

Microsoft Corp. Windows NT Server Product Documentation: Administration Kit Guide—Connection Manager, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cmak.mspx (Connection Manager) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp. Autodial Heuristics, available at http://support.microsoft.com/kb/164249 (Autodial Heuristics) (Although undated, this reference refers to the operation of prior art versions of Microsoft

Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Cariplo: Distributed Component Object Model, (1996) available at http://msdn2.microsoft.com/en-us/library/ms809332(printer).aspx (Cariplo I).

Marc Levy, COM Internet Services (Apr. 23, 1999), available at http://msdn2.microsoft.com/en-us/library/ms809302(printer).aspx (Levy).

Markus Horstmann and Mary Kirtland, DCOM Architecture (Jul. 23, 1997), available at http://msdn2.microsoft.com/en-us/library/ms809311(printer).aspx (Horstmann).

Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at http://msdn2.microsoft.com/en-us/library/ms809320(printer).aspx (DCOM Business Overview I).

Microsoft Corp., DCOM Technical Overview (Nov. 1996), available at http://msdn2.microsoft.com/en-us/library/ms809340(printer).aspx (DCOM Technical Overview I).

Microsoft Corp., DCOM Architecture White Paper (1998) available in PDC DVD-ROM (DCOM Architecture).

Microsoft Corp, DCOM—The Distributed Component Object Model, A Business Overview White Paper (Microsoft 1997) available in PDC DVD-ROM (DCOM Business Overview II).

Microsoft Corp., DCOM—Cariplo Home Banking Over The Internet White Paper (Microsoft 1996) available in PDC DVD-ROM (Cariplo II).

Microsoft Corp., DCOM Solutions in Action White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Solutions in Action).

Microsoft Corp., DCOM Technical Overview White Paper (Microsoft 1996) available 12 in PDC DVD-ROM (DCOM Technical Overview II).

Scott Suhy & Glenn Wood, DNS and Microsoft Windows NT 4.0, (1996) available at http://msdn2.microsoft.com/en-us/library/ms810277(printer).aspx (Suhy).

Aaron Skonnard, *Essential Winlnet* 313-423 (Addison Wesley Longman 1998) (Essential Winlnet).

Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at http://msdn2.microsoft.com/enus/library/ms811078(printer).aspx (Using PPTP).

Microsoft Corp., Internet Connection Services for MS RAS, Standard Edition, http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstart.mspx (Internet Connection Services I).

Microsoft Corp., Internet Connection Services for RAS, Commercial Edition, available athttp://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspx (Internet Connection Services II).

Microsoft Corp., Internet Explorer 5 Corporate Deployment Guide—Appendix B:Enabling Connections with the Connection Manager Administration Kit, available at http://www.microsoft.com/technet/prodtechnol/ ie/deploy/deploy5/appendb.mspx (IE5 Corporate Development).

Mark Minasi, *Mastering Windows NT Server 4* 1359-1442 (6th ed., Jan. 15, 1999)(Mastering Windows NT Server).

Hands On, Self-Paced Training for Supporting Version 4.0 371-473 (Microsoft Press 1998) (Hands On).

Microsoft Corp., MS Point-to-Point Tunneling Protocol (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/maintain/featusability/pptpwp3.mspx (MS PPTP).

Kenneth Gregg, et al., *Microsoft Windows NT Server Administrator's Bible* 173-206, 883-911, 974-1076 (IDG Books Worldwide 1999) (Gregg).

Microsoft Corp., Remote Access (Windows), available at http://msdn2.microsoft.com/en-us/library/bb545687(VS.85,printer).aspx (Remote Access).

Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspx (Understanding PPTP NT 4) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Windows NT 4.0: Virtual Private Networking, available at http://www.microsoft.com/technet/archive/winntas/ deploy/confeat/vpntwk.mspx (NT4 VPN) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Anthony Northrup, *NT Network Plumbing: Routers, Proxies, and Web Services* 299-399 (IDG Books Worldwide 1998) (Network Plumbing).

Microsoft Corp., Chapter 1—Introduction to Windows NT Routing with Routing and Remote Access Service, Available at http://www.microsoft.com/technet/archive/winntas/proddocs/ rras40/rrasch01.mspx (Intro to RRAS) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.) 13.

Microsoft Corp., Windows NT Server Product Documentation: Chapter 5—Planning for Large-Scale Configurations, available at http://www.microsoft.com/technet/archive/winntas/proddocs/rras40/rrasch05.mspx (Large-Scale Configurations) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

F-Secure, *F-Secure NameSurfer* (May 1999) (from FSECURE 00000003) (NameSurfer 3).

F-Secure, *F-Secure VPN Administrator's Guide* (May 1999) (from FSECURE 00000003) (F-Secure VPN 3).

F-Secure, *F-Secure SSH User's & Administrator's Guide* (May 1999) (from FSECURE 00000003) (SSH Guide 3).

F-Secure, *F-Secure SSH2.0 for Windows NT and 95* (May 1999) (from FSECURE 00000003) (SSH 2.0 Guide 3).

F-Secure, *F-Secure VPN+ Administrator's Guide* (May 1999) (from Fsecure 00000003) (VPN+ Guide 3).

F-Secure, *F-Secure VPN+ 4.1* (1999) (from Fsecure 00000006) (VPN+ 4.1 Guide 6).

F-Secure, *F-Secure SSH* (1996) (from Fsecure 00000006) (F-Secure SSH 6).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (1998) (from Fsecure 00000006) (F-Secure SSH 2.0 Guide 6).

F-Secure, *F-Secure SSH User's & Administrator's Guide* (Sep. 1998) (from Fsecure 00000009) (SSH Guide 9).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (Sep. 1998) (from Fsecure 00000009) (F-Secure SSH 2.0 Guide 9).

F-Secure, *F-Secure VPN+* (Sep. 1998) (from Fsecure 00000009) (VPN+ Guide 9).

F-Secure, *F-Secure Management Tools, Administrator's Guide* (1999) (from Fsecure 00000003) (F-Secure Management Tools).

F-Secure, *F-Secure Desktop, User's Guide* (1997) (from Fsecure 00000009) (FSecure Desktop User's Guide).

SafeNet, Inc., *VPN Policy Manager* (Jan. 2000) (VPN Policy Manager).

F-Secure, F-Secure VPN+ for Windows NT 4.0 (1998) (from Fsecure 00000009) (FSecure VPN+).

IRE, Inc., *SafeNet / Security Center Technical Reference Addendum* (Jun. 22, 1999) (Safenet Addendum).

IRE, Inc., *System Description for VPN Policy Manager and SafeNet/SoftPK* (Mar. 30, 2000) (VPN Policy Manager System Description).

IRE, Inc., About SafeNet / VPN Policy Manager (1999) (About Safenet VPN Policy Manager).

Trusted Information Systems, Inc., *Gauntlet Internet Firewall, Firewall Product Functional Summary* (Jul. 22, 1996) (Gauntlet Functional Summary).

Trusted Information Systems, Inc., *Running the Gauntlet Internet Firewall, An Administrator's Guide to Gauntlet Version 3.0* (May 31, 1995) (Running the Gauntlet Internet Firewall).

Ted Harwood, *Windows NT Terminal Server and Citrix Metaframe* (New Riders 1999) (Windows NT Harwood) 79.

Todd W. Matehrs and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implemetning Terminal Server and Citrix MetaFrame* (Macmillan Technial Publishing 1999) (Windows NT Mathers).

Bernard Aboba et al., *Securing L2TP using IPSEC* (Feb. 2, 1999).

Finding Your Way Through the VPN Maze (1999) ("PGP").

Linux FreeS/WAN Overview (1999) (Linux FreeS/WAN) Overview).

TimeStep, *The Business Case for Secure VPNs* (1998) ("TimeStep").

WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).

WatchGuard Technologies, Inc., *MSS Firewall Specifications* (1999).

WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).

WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).

WatchGuard Technologies, Inc., *WatchGuard LiveSecurity for MSS Powerpoint* (Feb. 14, 2000).

WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Releaset Notes* (Jul. 21, 2000).

Air Force Research Laboratory, *Statement of Work for Information Assurance System Architecture and Integration*, PR No. N-8-6106 (Contract No. F30602-98-C-0012) (Jan. 29, 1998).

GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.2 Report, Rev. 1.0* (Sep. 21, 1998).

BBN Information Assurance Contract, *TIS Labs Monthly Status Report* (Mar. 16-Apr. 30, 1998).

DARPA, *Dynamic Virtual Private Network (VPN) Powerpoint*.

GTE Internetworking, *Contractor's Program Progress Report* (Mar. 16-Apr. 30, 1998).

Darrell Kindred, *Dynamic Virtual Private Networks (DVPN) Countermeasure Characterization* (Jan. 30, 2001).

Virtual Private Networking Countermeasure Characterization (Mar. 30, 2000).

Virtual Private Network Demonstration (Mar. 21, 1998).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks (VPNs) and Integrated Security Management* (2000).

Information Assurance/NAI Labs, *Create/Add DVPN Enclave* (2000).

NAI Labs, *IFE 3.1 Integration Demo* (2000).

Information Assurance, *Science Fair Agenda* (2000).

Darrell Kindred et al., *Proposed Threads for IFE 3.1* (Jan. 13, 2000).

IFE 3.1 Technology Dependencies (2000).

IFE 3.1 Topology (Feb. 9, 2000).

Information Assurance, *Information Assurance Integration: IFE 3.1, Hypothesis & Thread Development* (Jan. 10-11, 2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation* (2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.2* (2000).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).

T. Braun et al., *Virtual Private Network Architecture*, Charging and Accounting Technology for the Internet (Aug. 1, 1999) (VPNA).

Network Associates Products—*PGP Total Network Security Suite, Dynamic Virtual Private Networks* (1999).

Microsoft Corporation, Microsoft Proxy Server 2.0 (1997) (Proxy Server 2.0, Microsoft Prior Art VPN Technology).

David Johnson et. al., *A Guide To Microsoft Proxy Server 2.0* (1999) (Johnson, Microsoft Prior Art VPN Technology).

Microsoft Corporation, *Setting Server Parameters* (1997 (copied from Proxy Server 2.0 CD labeled MSFTVX00157288) (Setting Server Parameters, Microsoft Prior Art VPN Technology).

Kevin Schuler, *Microsoft Proxy Server 2* (1998) (Schuler, Microsoft Prior Art VPN Technology).

Erik Rozell et. al., *MCSE Proxy Server 2 Study Guide* (1998) (Rozell, Microsoft Prior 15 Art VPN Technology).

M. Shane Stigler & Mark A Linsenbardt, *IIS 4 and Proxy Server 2* (1999) (Stigler, Microsoft Prior Art VPN Technology).

David G. Schaer, *MCSE Test Success: Proxy Server 2* (1998) (Schaer, Microsoft Prior Art VPN Technology).

John Savill, *The Windows NT and Windows 2000 Answer Book* (1999) (Savill, Microsoft Prior Art VPN Technology).

Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).

File History for U.S. Appl. No. 09/653,201, Applicant(s): Whittle Bryan, et al., filed Aug. 31, 2000.

AutoSOCKS v2.1, Datasheet, http://web.archive.org/web/19970212013409/www.aventail.com/prod/autoskds.html.

Ran Atkinson, *Use of DNS to Distribute Keys*, Sep. 7, 1993, http://ops.ietf.org/lists/namedroppers/namedroppers.199x/msg00945.html.

FirstVPN Enterprise Networks, Overview.

Chapter 1: Introduction to Firewall Technology, Administration Guide; Dec. 19, 2007, http://www.books24x7.com/book/id_762/viewer_r.asp?bookid=762&chunked=41065062.

The TLS Protocol Version 1.0; Jan. 1999; p. 65 of 71.

Elizabeth D. Zwicky, et al., Building Internet Firewalls, 2nd Ed.

Virtual Private Networks—Assured Digital Incorporated—ADI 4500; http://web.archive.org/web/19990224050035/www.assured-digital.com/products/prodvpn/adia4500.htm.

Accessware—The Third Wave in Network Security, Conclave from Internet Dynamics; http://web.archive.org/web/11980210013830/interdyn.com/Accessware.html.

Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks*, www.extendedsystems.com.

Socks Version 5; Executive Summary; http://web.archive.org/web/199706200319 45/www.aventail.com/educate/whitepaper/sockswp.html.

Internet Dynamics First to Ship Integrated Security Solutions for Enterprise Intranets and Extranets; Sep. 15, 1997; http://web.archive.org/web/19980210014150/interdyn.com.

Emails from various individuals to Linux IPsec re: DNS-LDAP Splicing.

Microsoft Corporation's Fifth Amended Invalidity Contentions dated Sep. 18, 2009, *VirnetX Inc. and Science Applications International Corp.* v. *Microsoft Corporation* and invalidity claim charts for U.S. Patent Nos. 7,188,180 and 6,839,759.

The IPSEC Protocol as described in Atkinson, et al., "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998) ("RFC 2401"); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and R. Glenn, "The Use HMAC-SHA-1-96 within ESP and AH," RFC 2404 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Derrell Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Douglas Maughan, et al, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

D. Harkins and D. Carrell, "The Internet Key Exchange (IKE)," RFC 2409 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec." RFC 2410 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

R. Thayer, et al., "IP Security Document Roadmap," RFC 2411 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

Hilarie K. Orman, "The OAKLEY Key Determination Protocol," RFC 2412 (Nov. 1998) in combination with J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose California (Jul. 1996) ("Galvin").

WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).

WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).

WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).

WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes* (Jul. 21, 2000).

Yuan Dong Feng, "A novel scheme combining interleaving technique with cipher in Rayleigh fading channels," Proceedings of the International Conference on Communication technology, 2:S47-02-1-S47-02-4 (1998).

D.W. Davies and W.L. Price, edited by Tadahiro Uezono, "Network Security", Japan, Nikkei McGraw-Hill, Dec. 5, 1958, First Edition, first copy, p. 102-108.

David Kosiur, "Building and Managing Virtual Private Networks" (1998).

P. Mockapetris, "Domain Names—Implementation and Specification," Network Working Group, RFC 1035 (Nov. 1987).

Request for *Inter Partes* Reexamination of Patent No. 6,502,135, dated Nov. 25, 2009.

Request for *Inter Partes* Reexamination of Patent No. 7,188,180, dated Nov. 25, 2009.

* cited by examiner

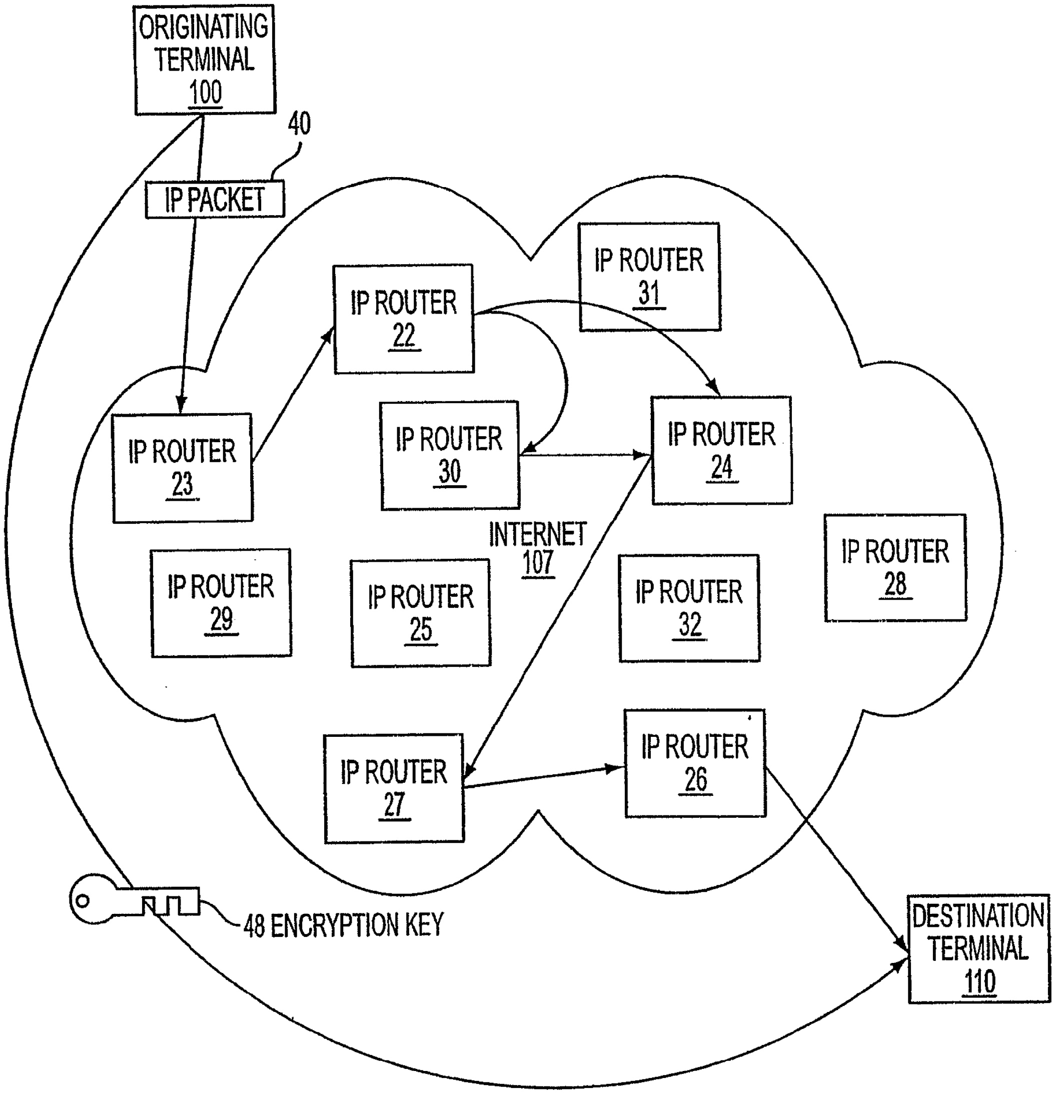

ORIGINATING TERMINAL 100
40
IP PACKET
IP ROUTER 22
IP ROUTER 31
IP ROUTER 30
IP ROUTER 24
IP ROUTER 23
INTERNET 107
IP ROUTER 29
IP ROUTER 25
IP ROUTER 32
IP ROUTER 28
IP ROUTER 27
IP ROUTER 26
48 ENCRYPTION KEY
DESTINATION TERMINAL 110

FIG. 1

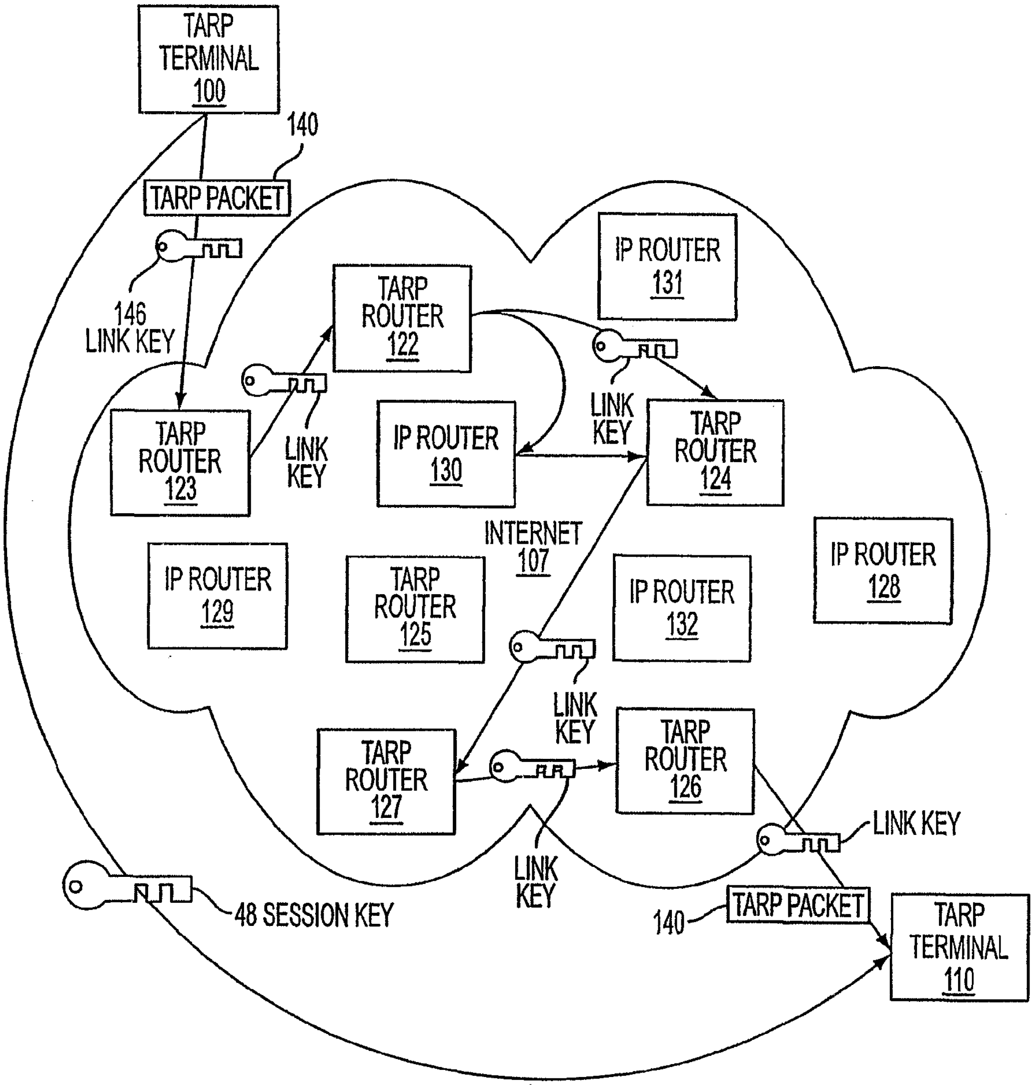

TARP TERMINAL 100
140
TARP PACKET
146 LINK KEY
TARP ROUTER 122
IP ROUTER 131
LINK KEY
TARP ROUTER 123
LINK KEY
IP ROUTER 130
TARP ROUTER 124
INTERNET 107
IP ROUTER 129
TARP ROUTER 125
IP ROUTER 132
IP ROUTER 128
LINK KEY
TARP ROUTER 127
TARP ROUTER 126
LINK KEY
LINK KEY
48 SESSION KEY
140
TARP PACKET
TARP TERMINAL 110

FIG. 2

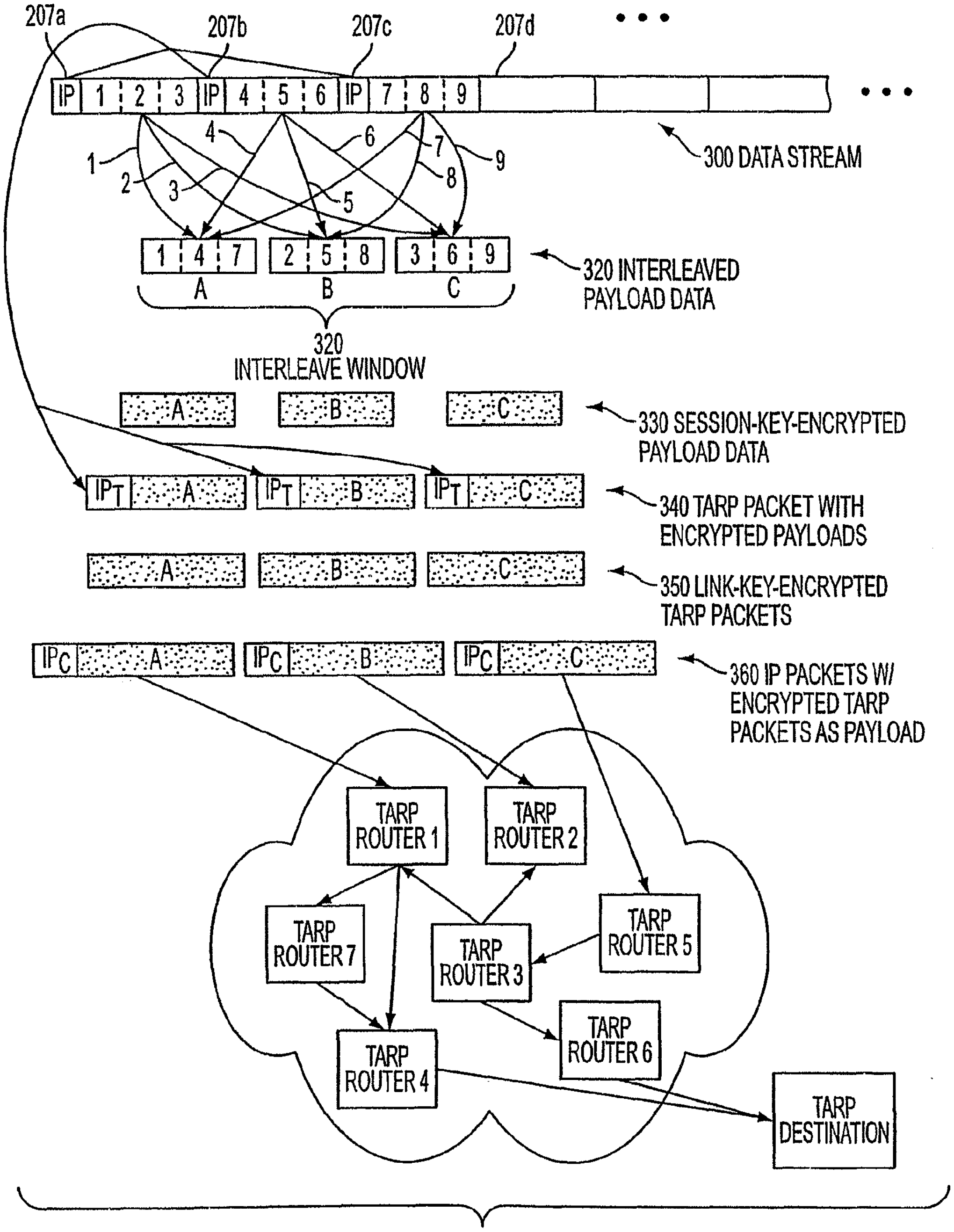

207a
207b
207c
207d
IP
1
2
3
IP
4
5
6
IP
7
8
9
1
2
3
4
5
6
7
8
9
300 DATA STREAM
1
4
7
2
5
8
3
6
9
A
B
C
320 INTERLEAVED PAYLOAD DATA
320 INTERLEAVE WINDOW
A
B
C
330 SESSION-KEY-ENCRYPTED PAYLOAD DATA
IP_T A
IP_T B
IP_T C
340 TARP PACKET WITH ENCRYPTED PAYLOADS
A
B
C
350 LINK-KEY-ENCRYPTED TARP PACKETS
IP_C A
IP_C B
IP_C C
360 IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 4
TARP ROUTER 6
TARP DESTINATION

FIG. 3A

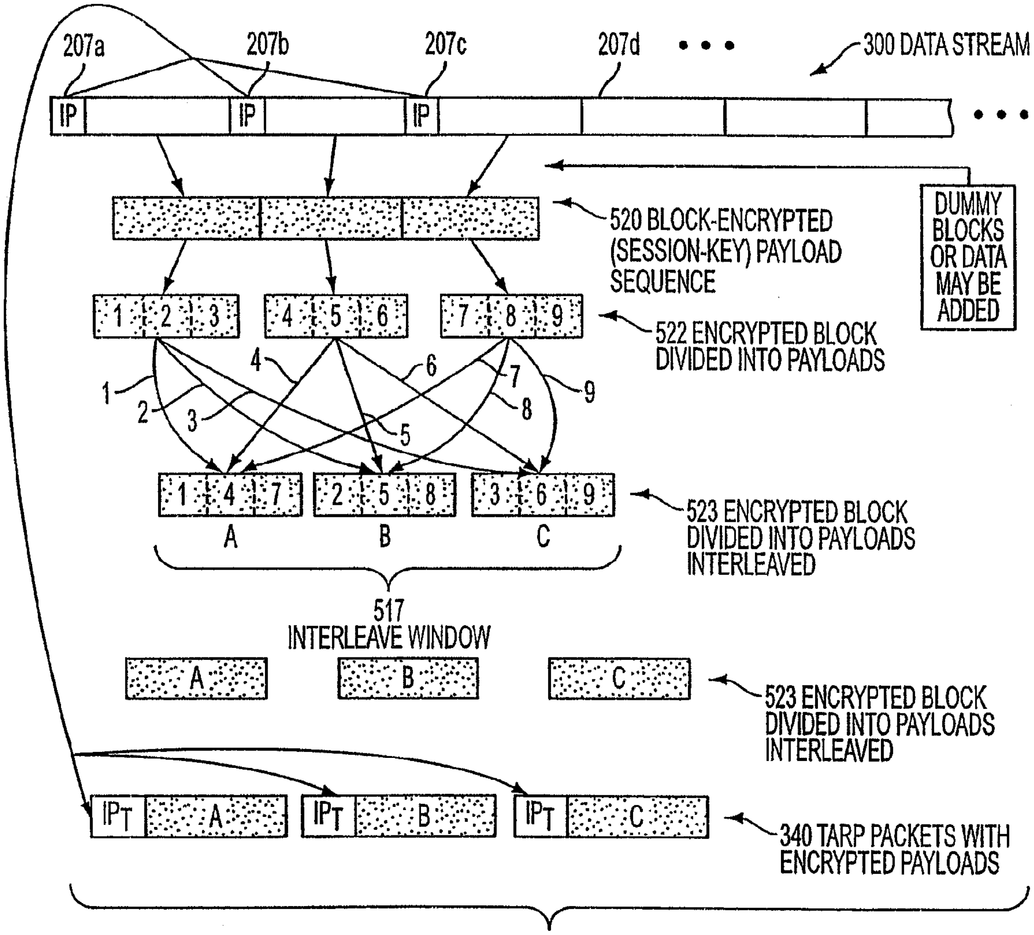

207a
207b
207c
207d
300 DATA STREAM
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
520 BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE
1 2 3
4 5 6
7 8 9
522 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS
1 2 3 4 5 6 7 8 9
1 4 7
2 5 8
3 6 9
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
517 INTERLEAVE WINDOW
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
IPT A
IPT B
IPT C
340 TARP PACKETS WITH ENCRYPTED PAYLOADS

FIG. 3B

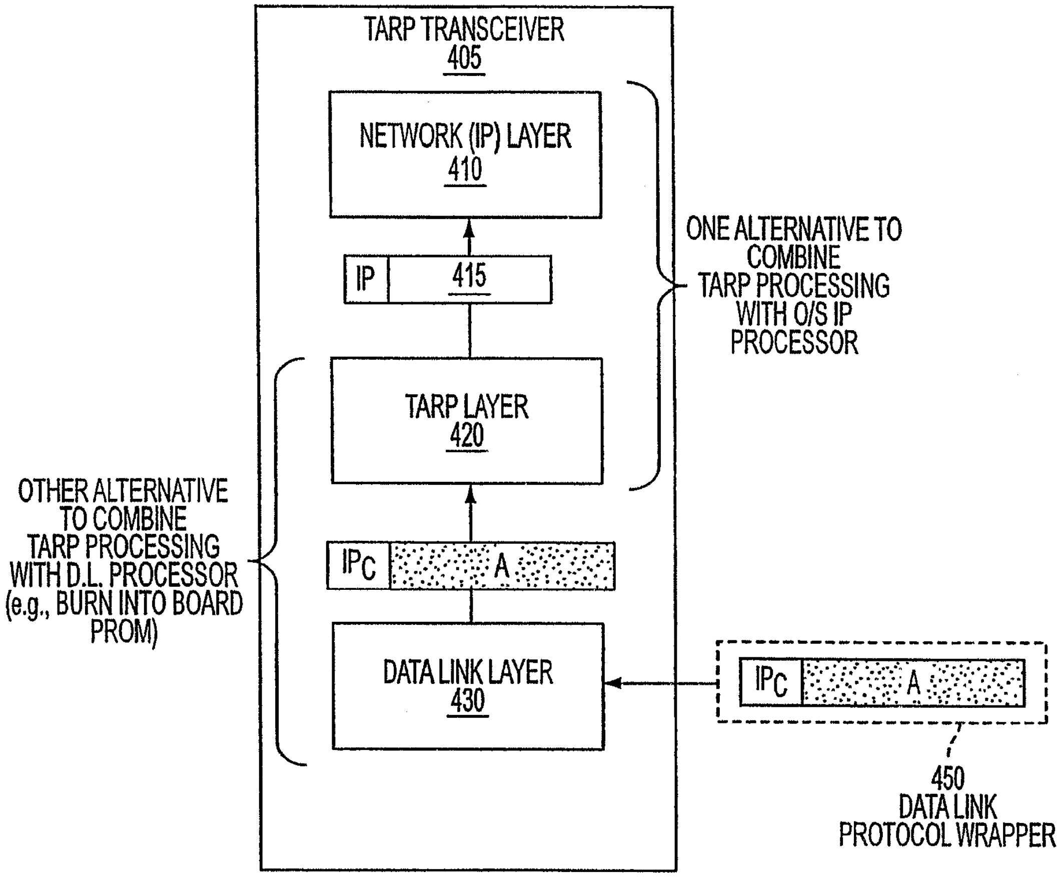

TARP TRANSCEIVER
405
NETWORK (IP) LAYER
410
IP
415
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
TARP LAYER
420
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (e.g., BURN INTO BOARD PROM)
IPC
A
DATA LINK LAYER
430
IPC
A
450
DATA LINK PROTOCOL WRAPPER

FIG. 4

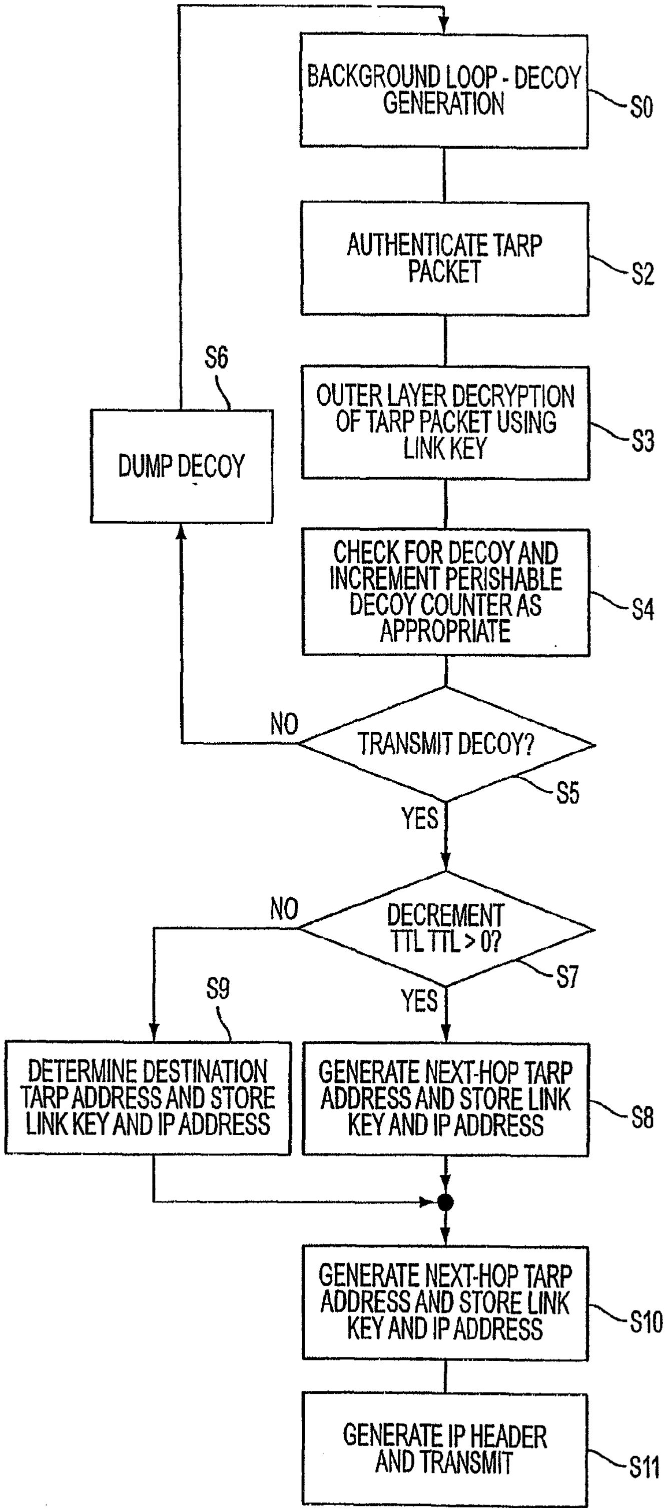

BACKGROUND LOOP - DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
S6
DUMP DECOY
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

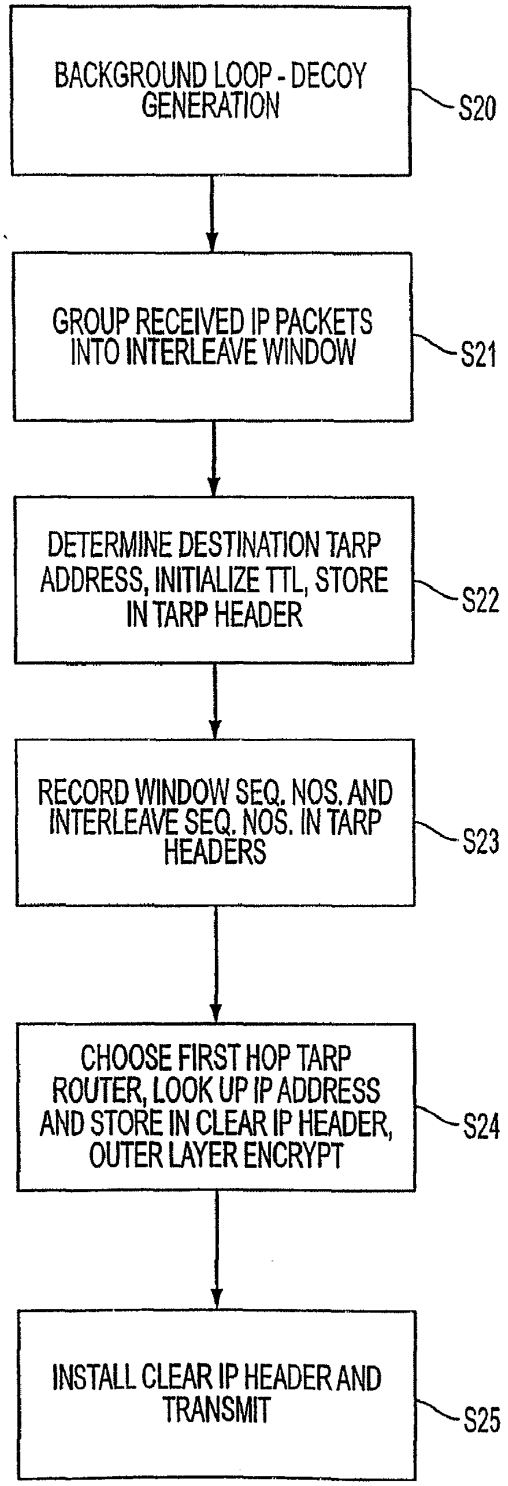

BACKGROUND LOOP - DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS. IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

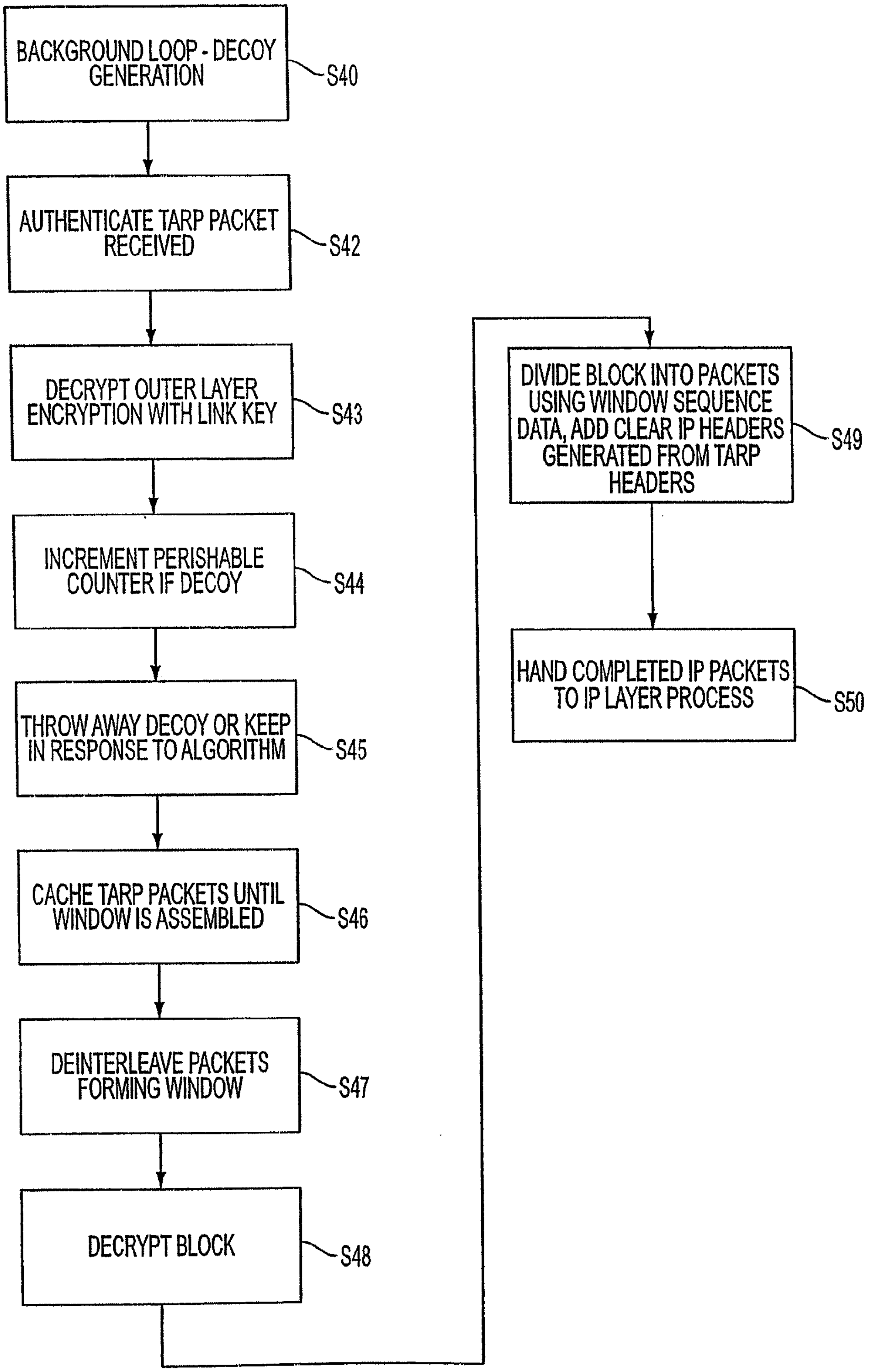

BACKGROUND LOOP - DECOY GENERATION
S40
AUTHENTICATE TARP PACKET RECEIVED
S42
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S43
INCREMENT PERISHABLE COUNTER IF DECOY
S44
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S45
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S46
DEINTERLEAVE PACKETS FORMING WINDOW
S47
DECRYPT BLOCK
S48
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S49
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS
S50

FIG. 7

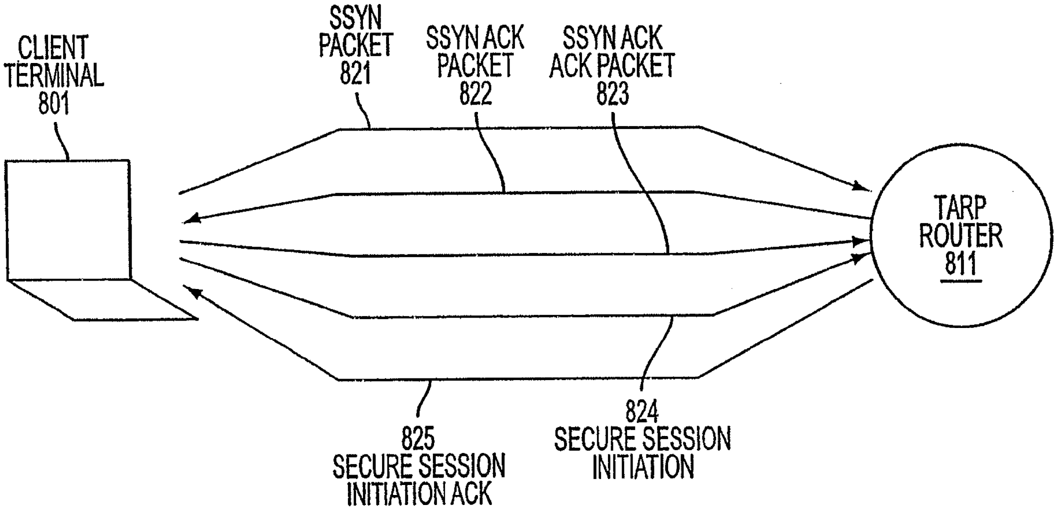
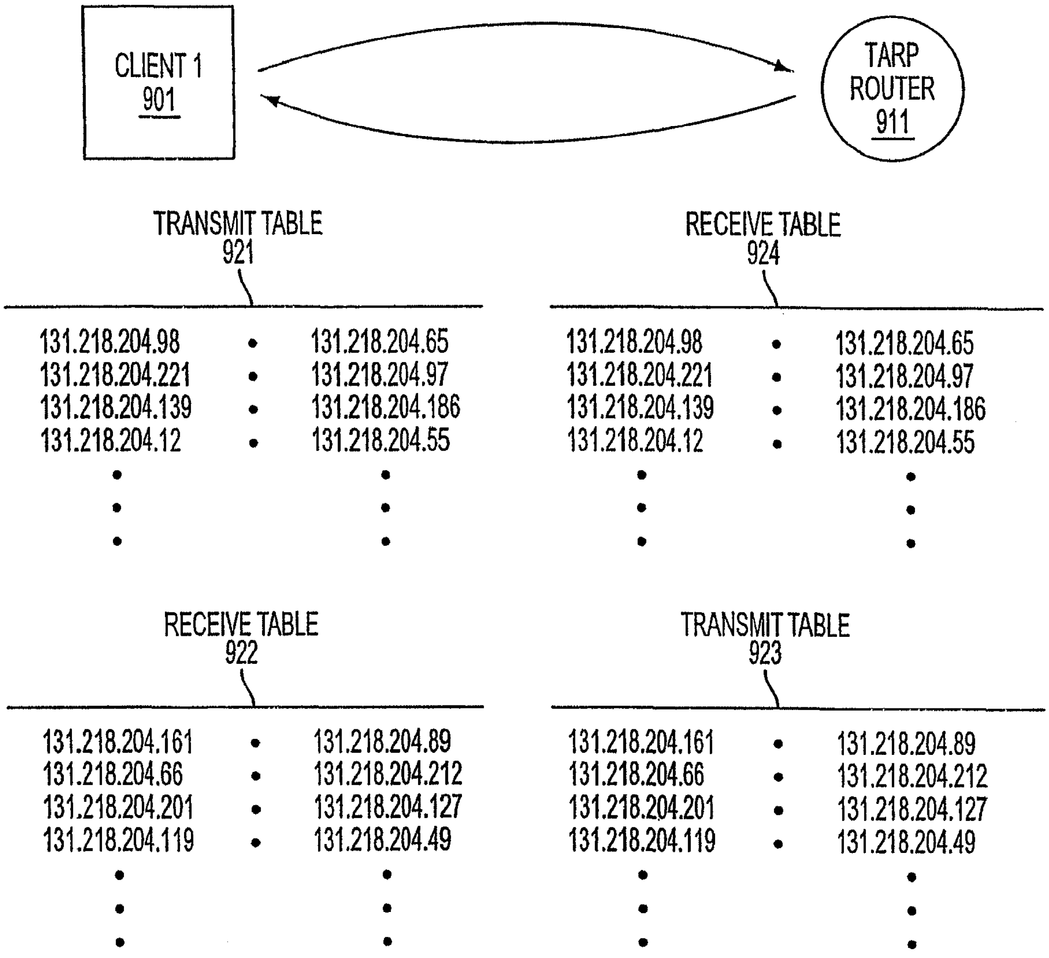

CLIENT TERMINAL 801
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
TARP ROUTER 811
825 SECURE SESSION INITIATION ACK
824 SECURE SESSION INITIATION

FIG. 8



TRANSMIT TABLE 921

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
•		•
•		•
•		•

RECEIVE TABLE 924

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
•		•
•		•
•		•

RECEIVE TABLE 922

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
•		•
•		•
•		•

TRANSMIT TABLE 923

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
•		•
•		•
•		•

FIG. 9

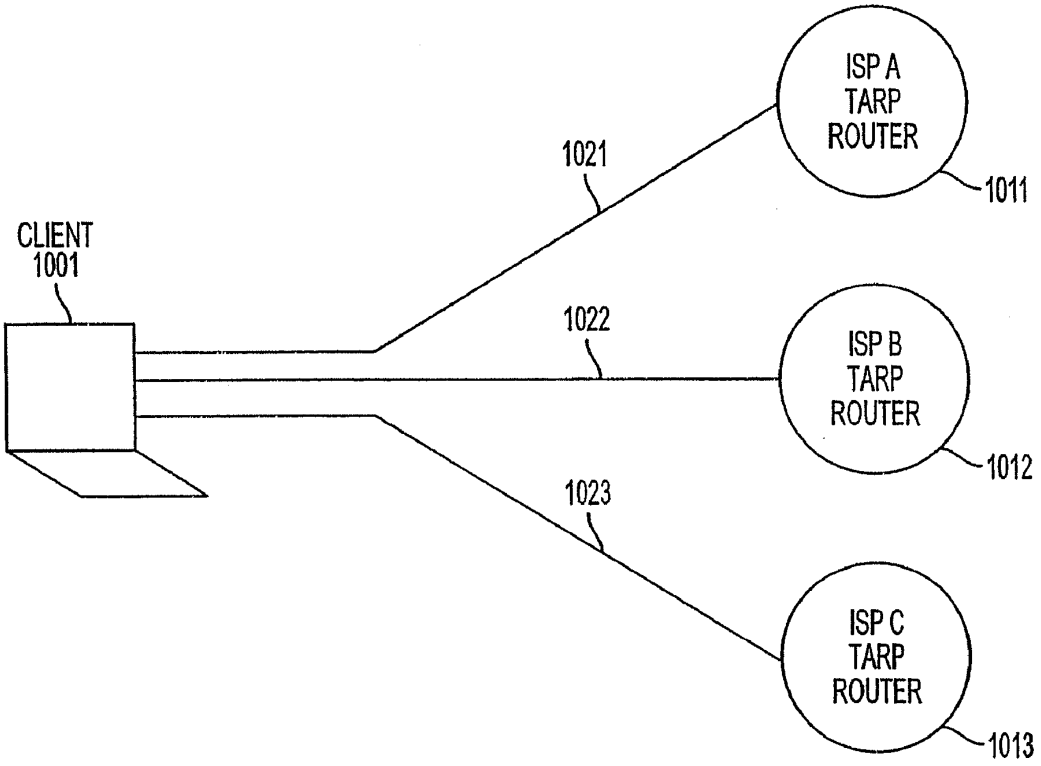

ISP A
TARP
ROUTER
1021
1011
CLIENT
1001
1022
ISP B
TARP
ROUTER
1012
1023
ISP C
TARP
ROUTER
1013

FIG. 10

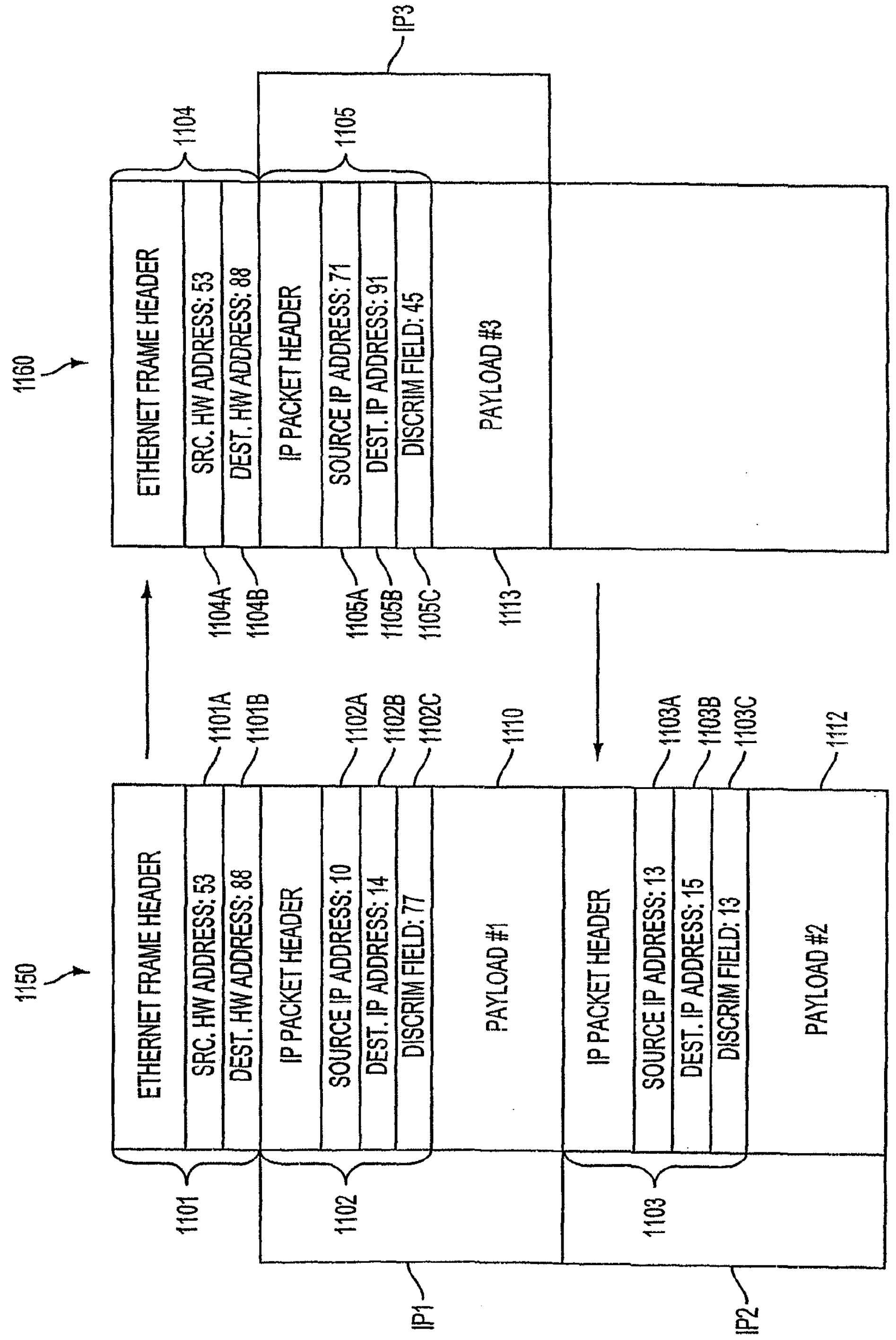
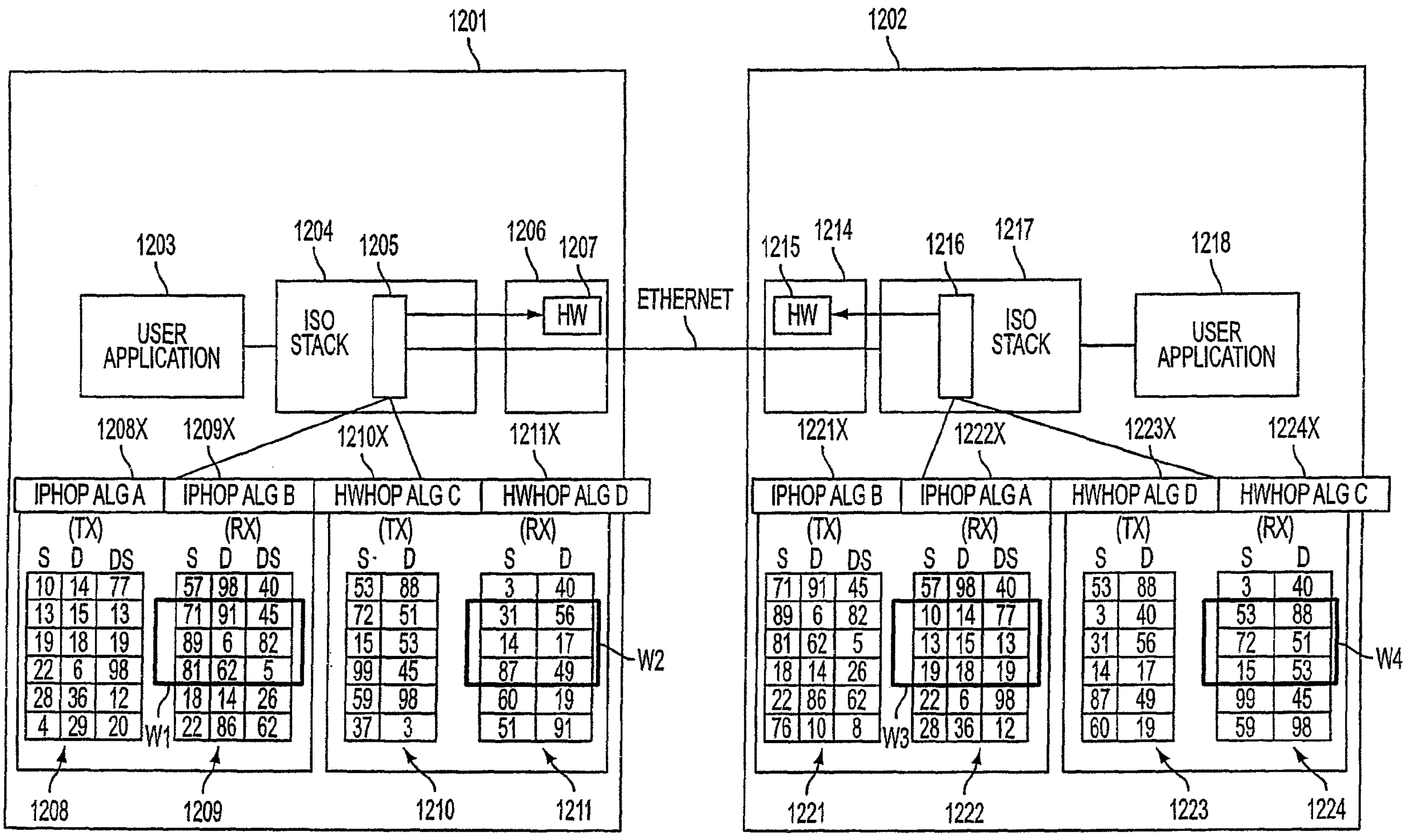

1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104
1105
IP3
1104A
1104B
1105A
1105B
1105C
1113
1150
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1101
1102
1103
IP1
IP2

FIG. 11



S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

S·	D
53	88
72	51
15	53
99	45
59	98
37	3

S	D
3	40
31	56
14	17
87	49
60	19
51	91

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

S	D
53	88
3	40
31	56
14	17
87	49
60	19

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

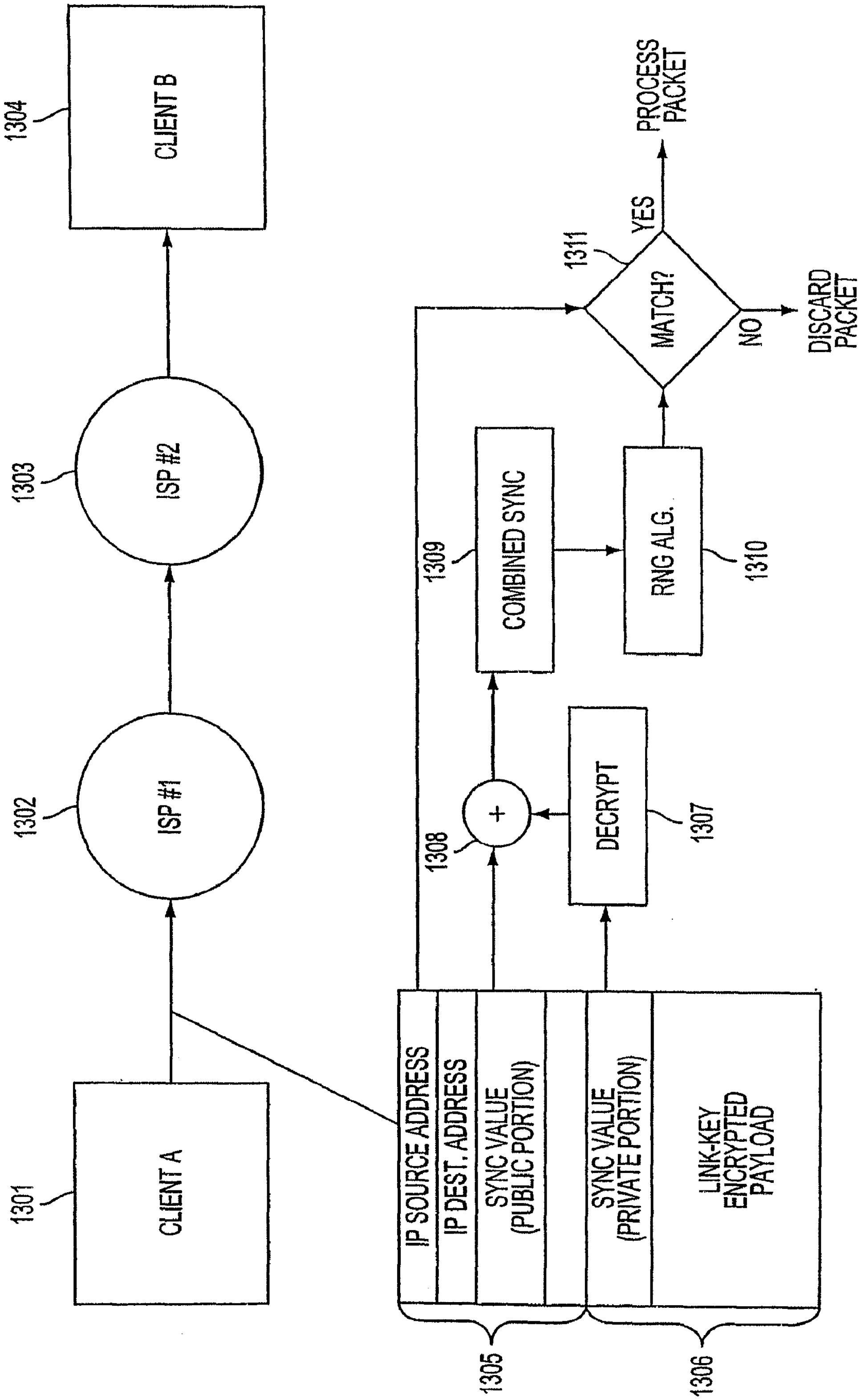

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
+
1308
COMBINED SYNC
1309
RNG ALG.
1310
MATCH?
1311
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

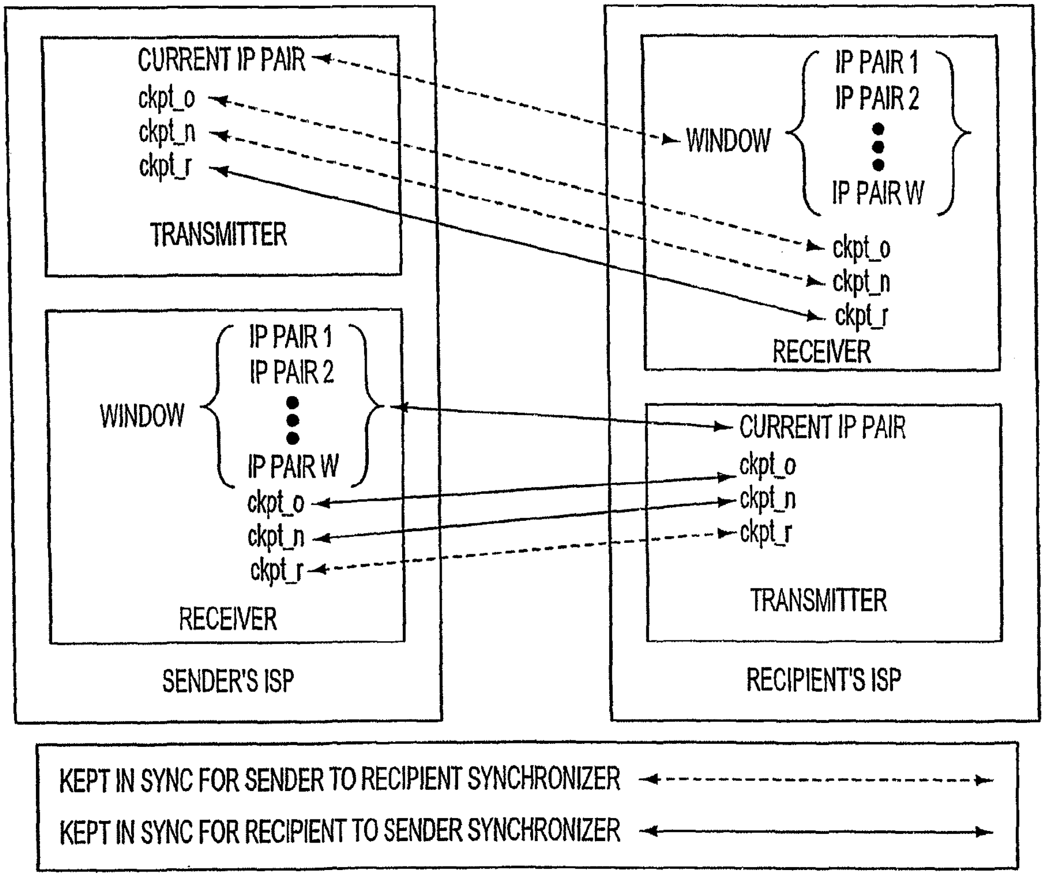

CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
SENDER'S ISP
RECIPIENT'S ISP
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

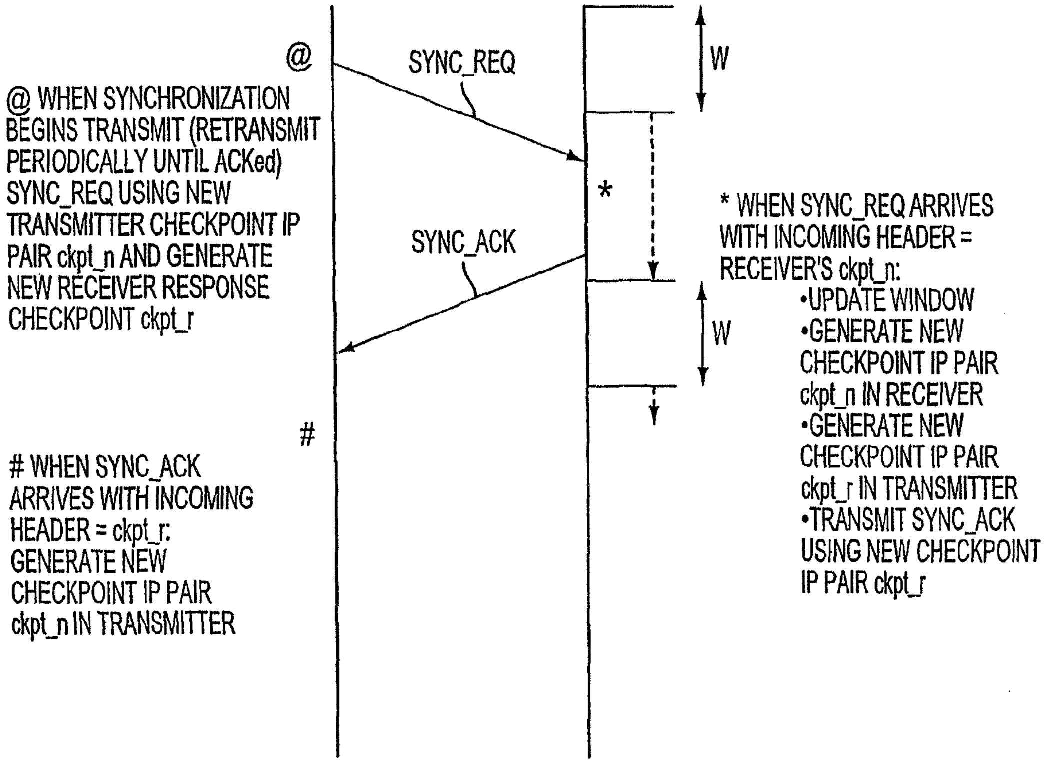

@
SYNC_REQ
W
@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
*
SYNC_ACK
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
•UPDATE WINDOW
•GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
•GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
•TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r
W
#
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

FIG. 15

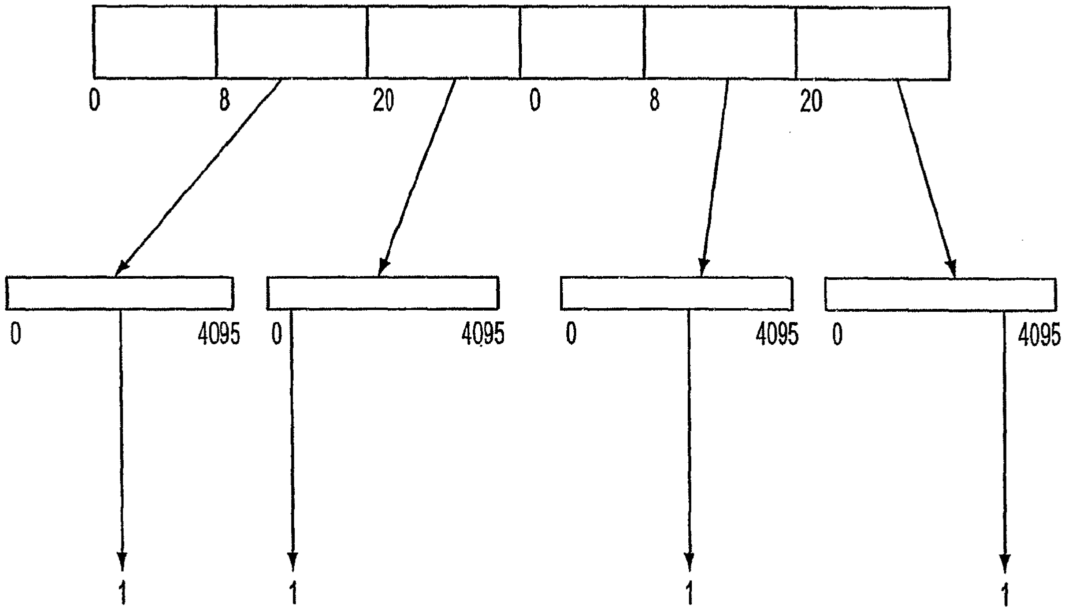


FIG. 16

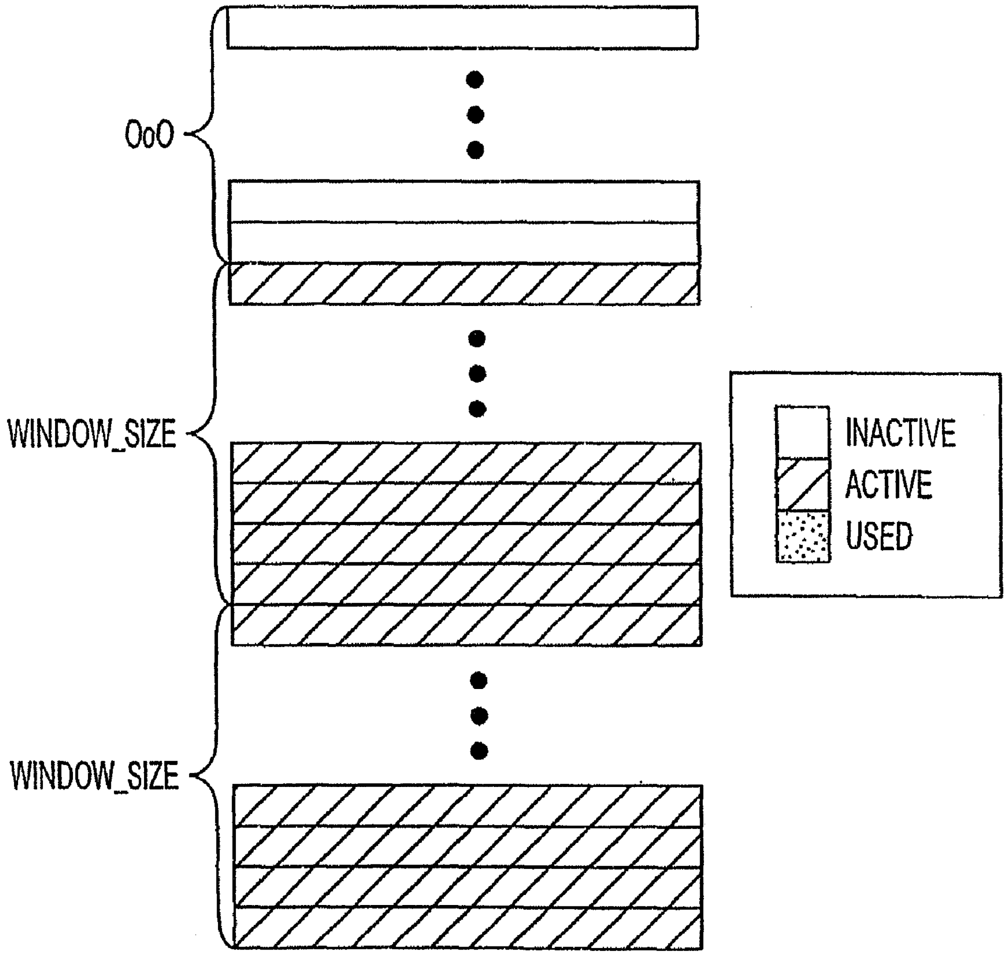


FIG. 17

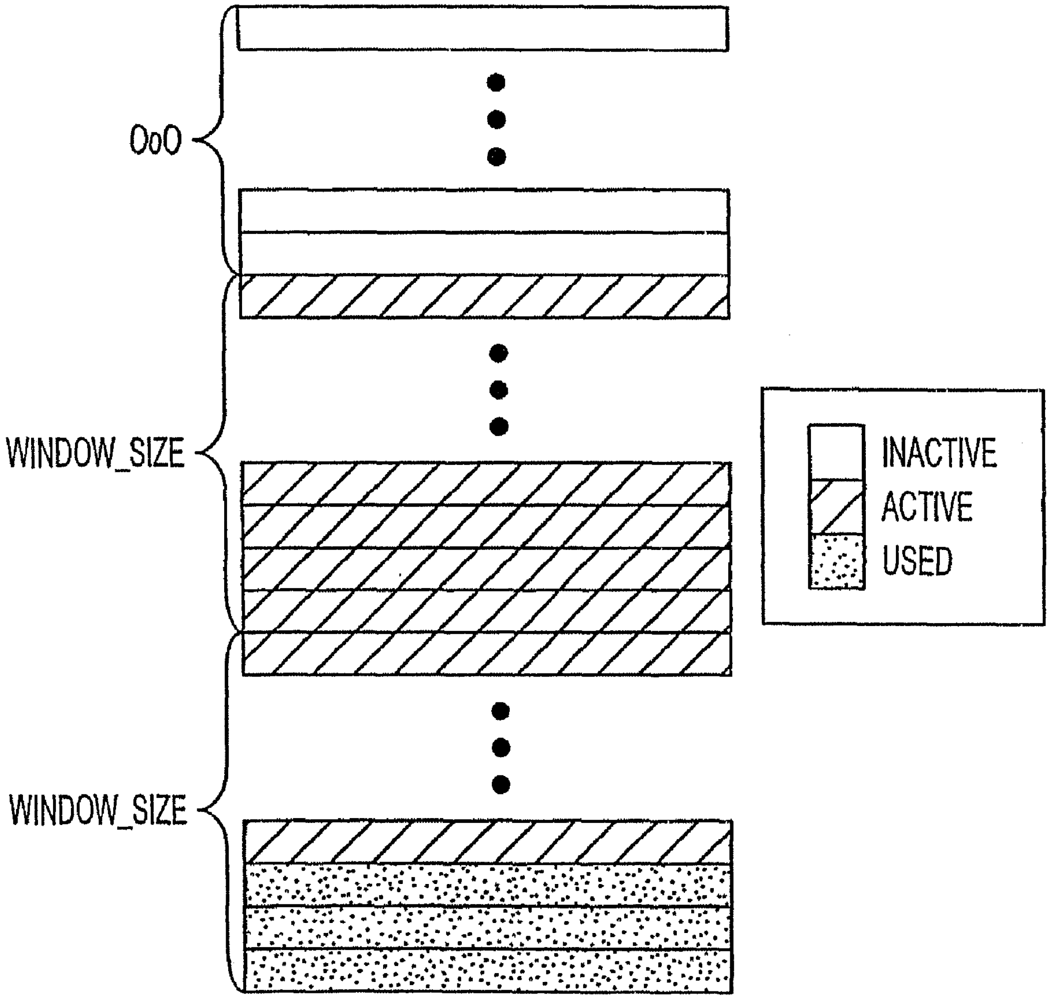


FIG. 18

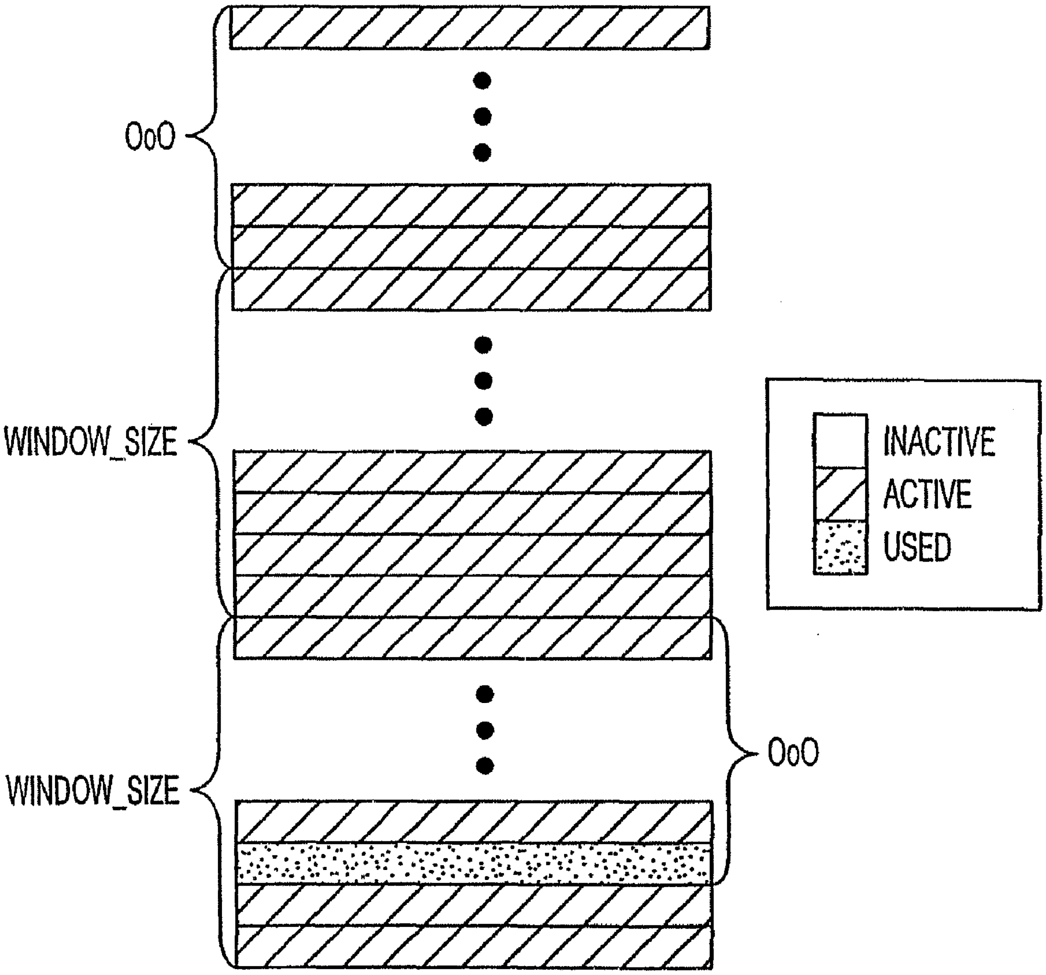


FIG. 19

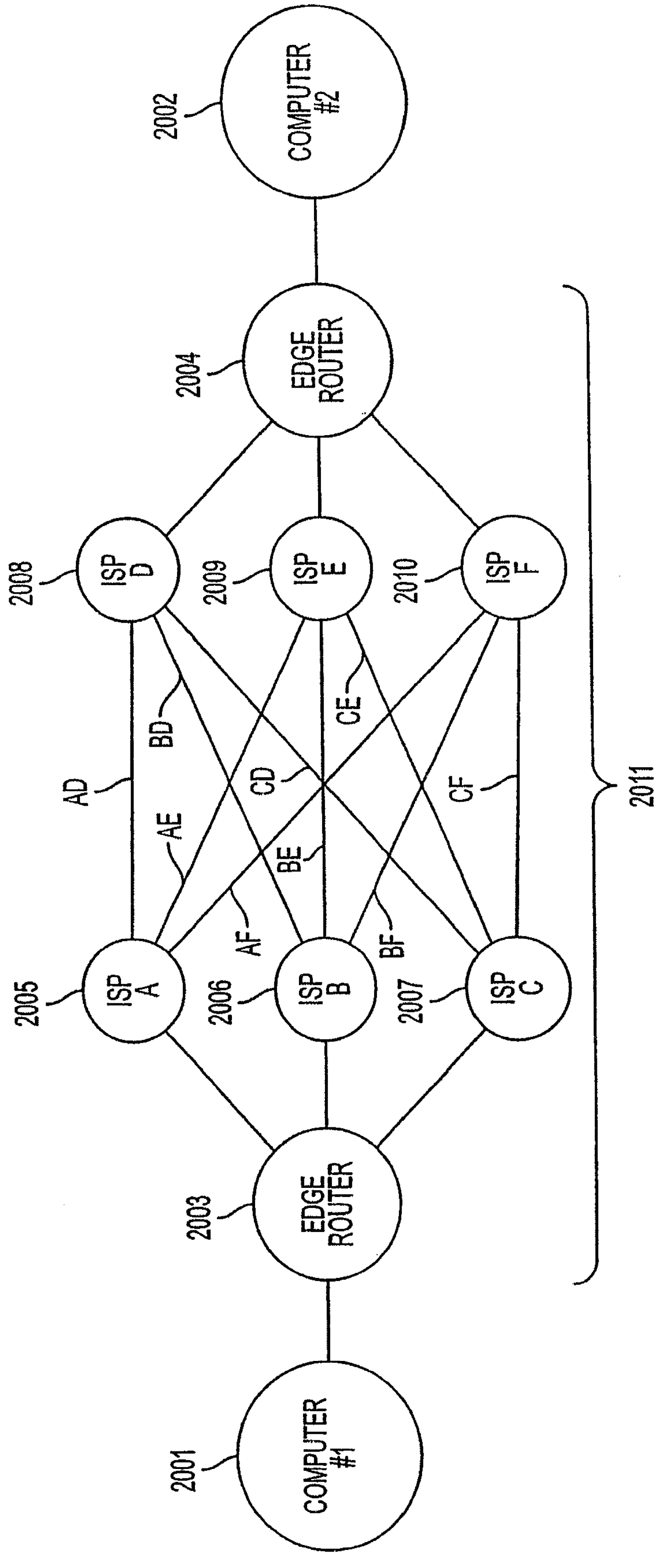

2002
COMPUTER #2
2004
EDGE ROUTER
2008
ISP D
2009
ISP E
2010
ISP F
AD
BD
CD
CE
AE
BE
CF
AF
BF
2005
ISP A
2006
ISP B
2007
ISP C
2003
EDGE ROUTER
2001
COMPUTER #1
2011

FIG. 20

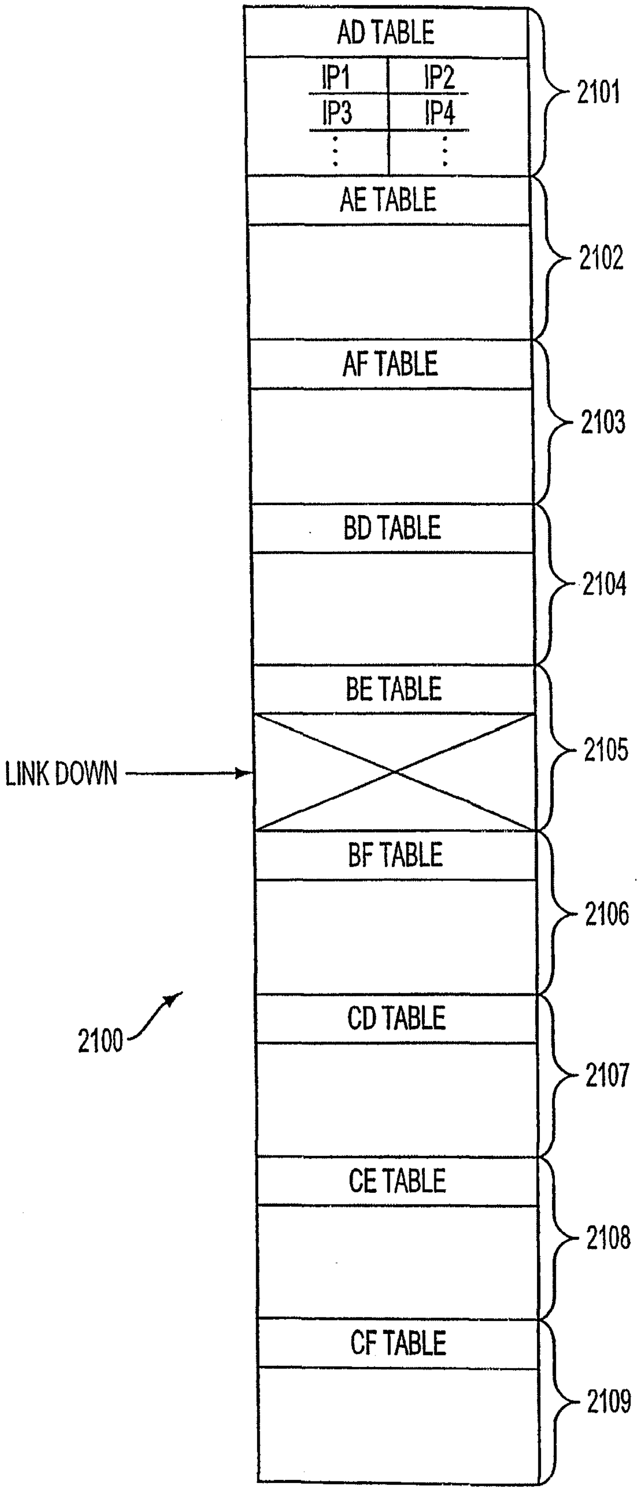

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

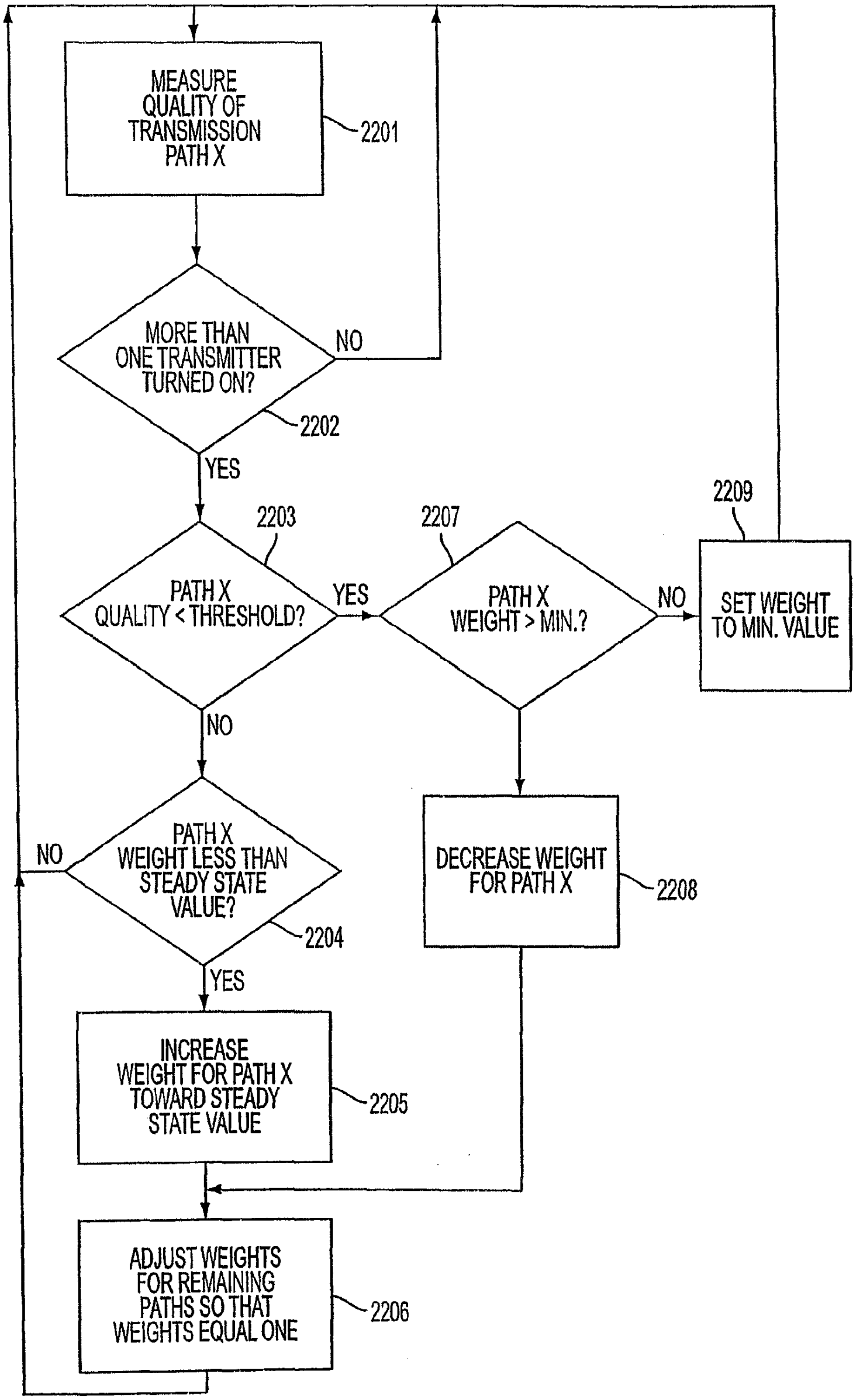

MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
NO
2202
YES
2203
PATH X QUALITY < THRESHOLD?
YES
2207
PATH X WEIGHT > MIN.?
NO
2209
SET WEIGHT TO MIN. VALUE
NO
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
2204
YES
DECREASE WEIGHT FOR PATH X
2208
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

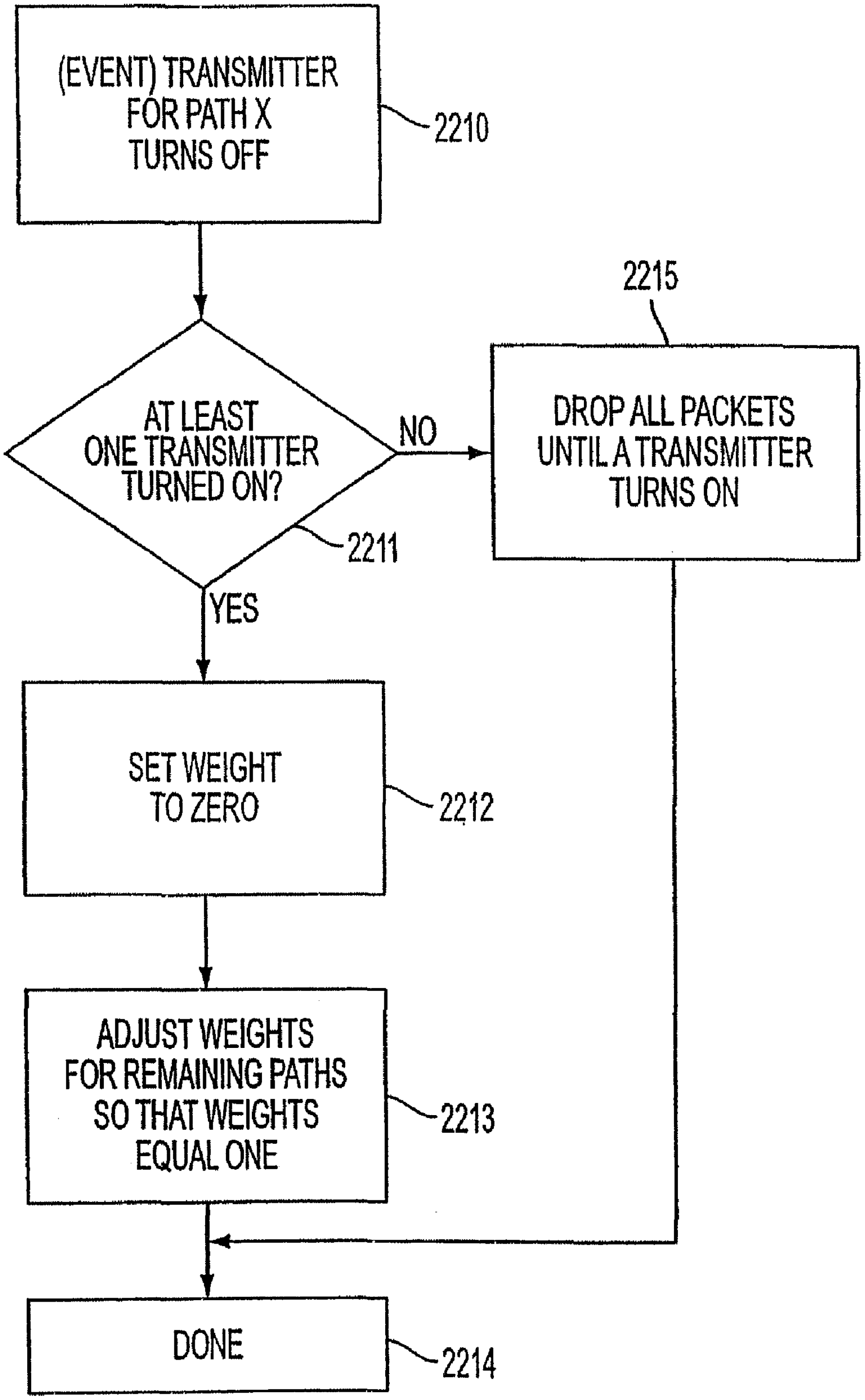

(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2210
AT LEAST ONE TRANSMITTER TURNED ON?
2211
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
SET WEIGHT TO ZERO
2212
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2213
DONE
2214

FIG. 22B

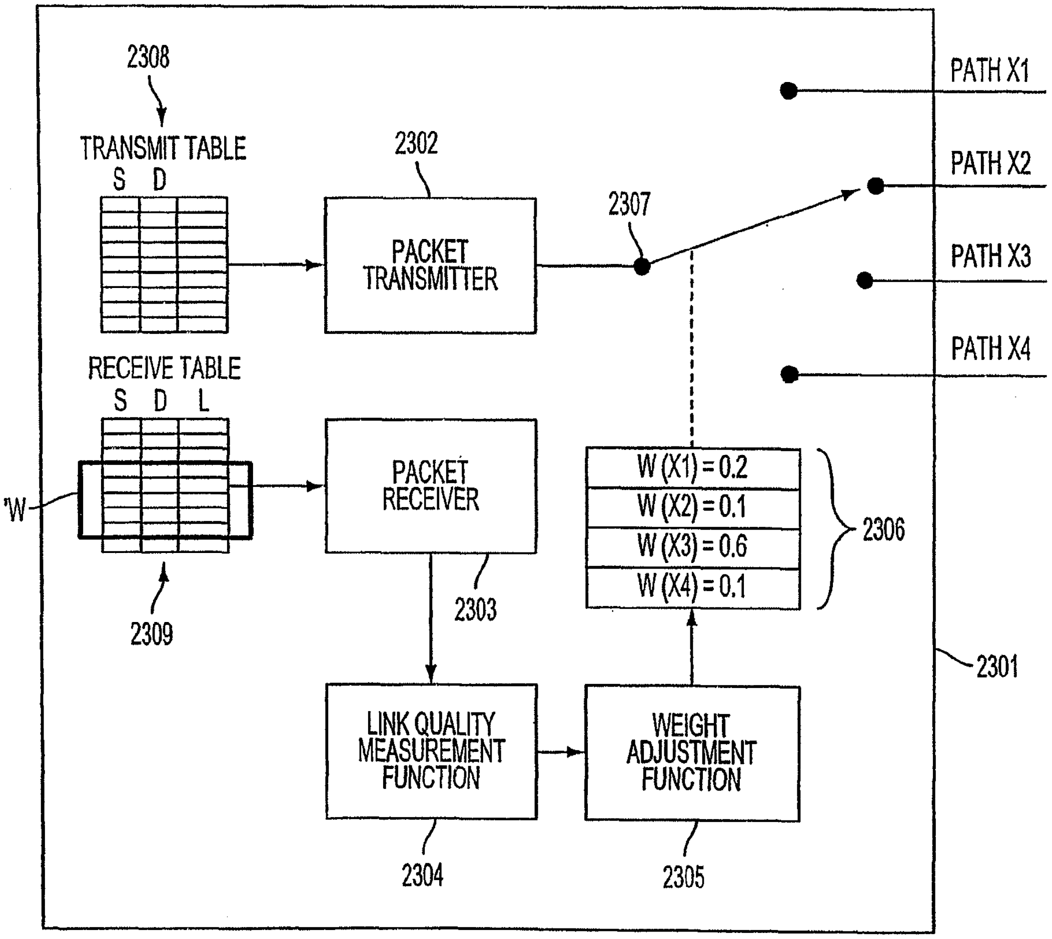

2308
TRANSMIT TABLE
S D
2302
PACKET TRANSMITTER
2307
PATH X1
PATH X2
PATH X3
PATH X4
RECEIVE TABLE
S D L
W
PACKET RECEIVER
W (X1) = 0.2
W (X2) = 0.1
W (X3) = 0.6
W (X4) = 0.1
2306
2309
2303
2301
LINK QUALITY MEASUREMENT FUNCTION
WEIGHT ADJUSTMENT FUNCTION
2304
2305

FIG. 23

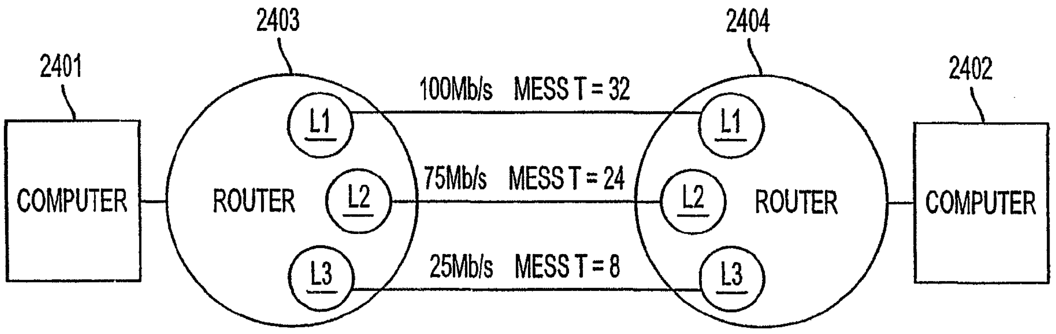

2401
2403
2404
2402
COMPUTER
ROUTER
L1
L2
L3
100Mb/s MESS T = 32
75Mb/s MESS T = 24
25Mb/s MESS T = 8
L1
L2
L3
ROUTER
COMPUTER

FIG. 24

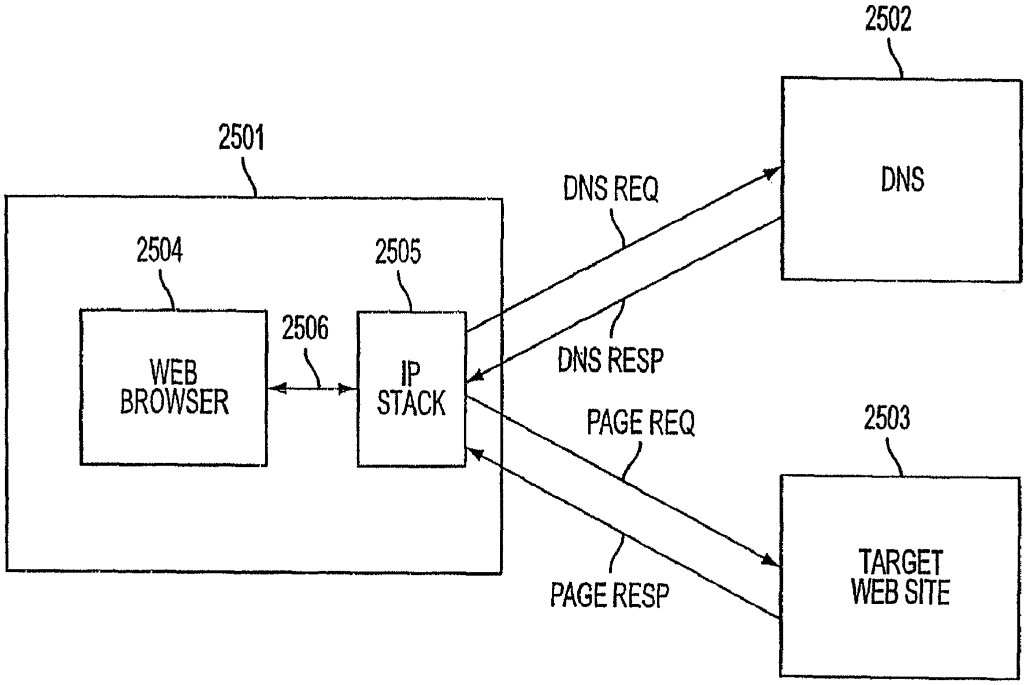


FIG. 25
(PRIOR ART)

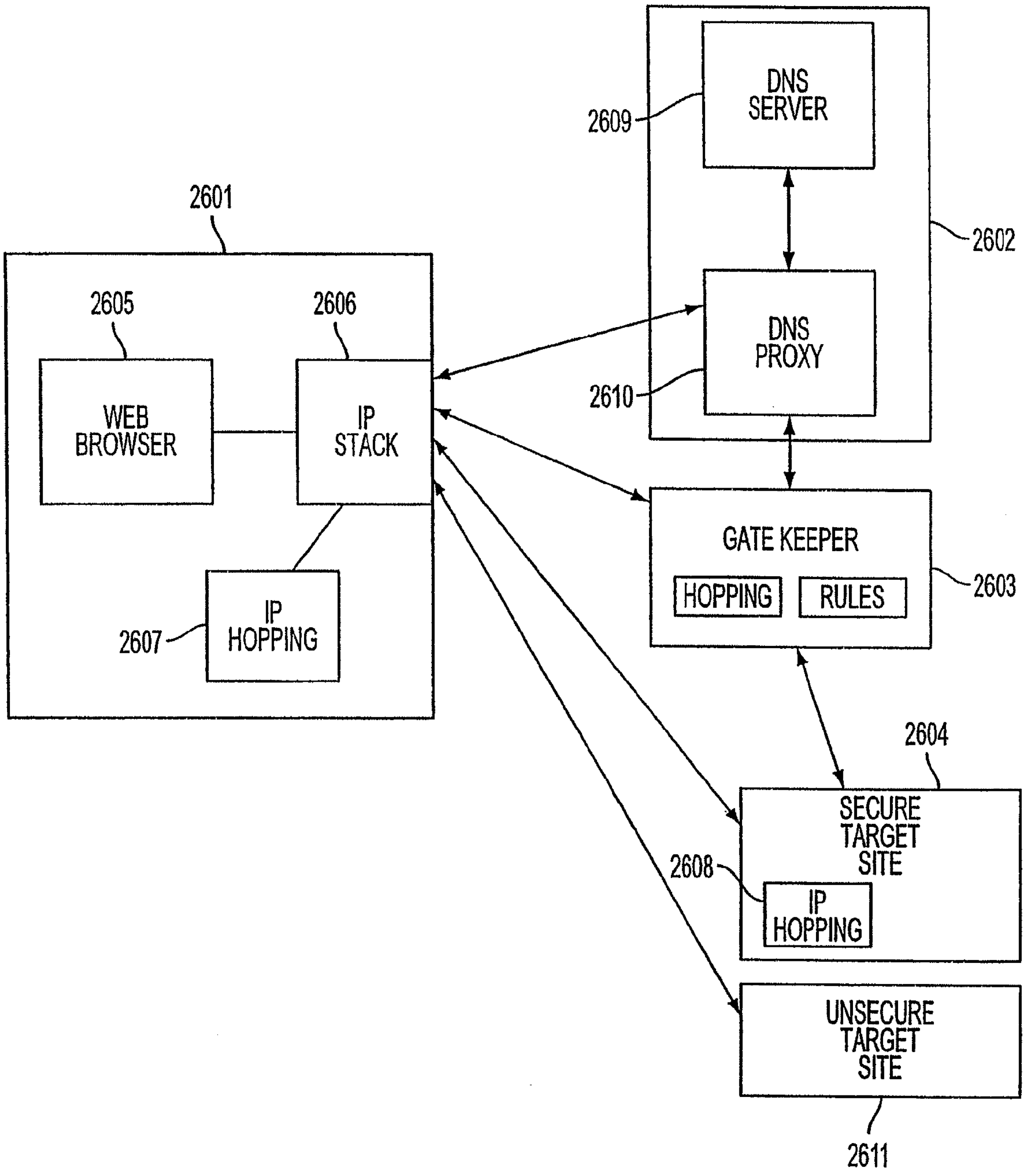

DNS SERVER
2609
2602
2601
2605
2606
DNS PROXY
WEB BROWSER
IP STACK
2610
GATE KEEPER
HOPPING
RULES
2603
IP HOPPING
2607
2604
SECURE TARGET SITE
2608
IP HOPPING
UNSECURE TARGET SITE
2611

FIG. 26

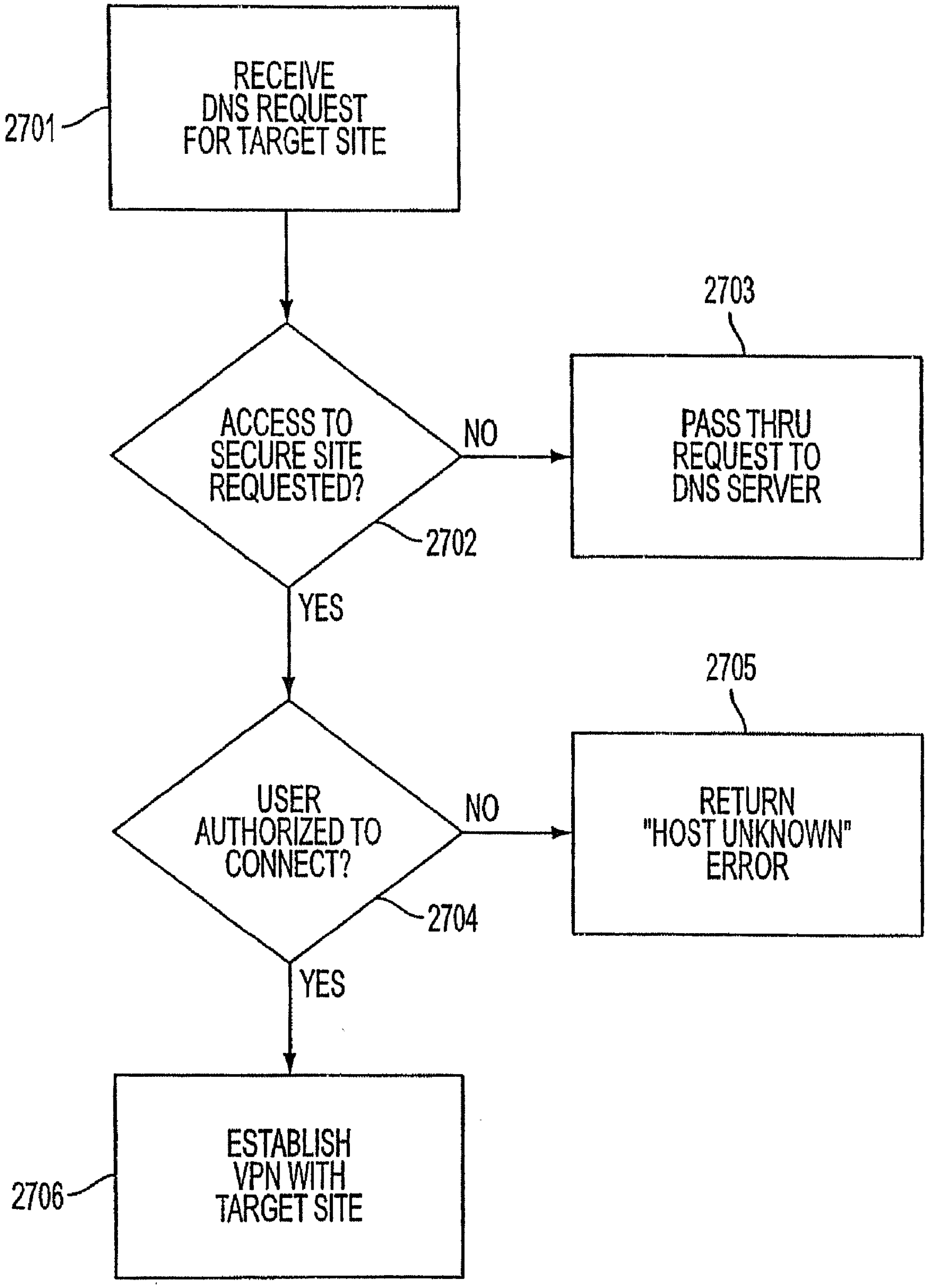

RECEIVE DNS REQUEST FOR TARGET SITE
2701
ACCESS TO SECURE SITE REQUESTED?
2702
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
USER AUTHORIZED TO CONNECT?
2704
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
ESTABLISH VPN WITH TARGET SITE
2706

FIG. 27

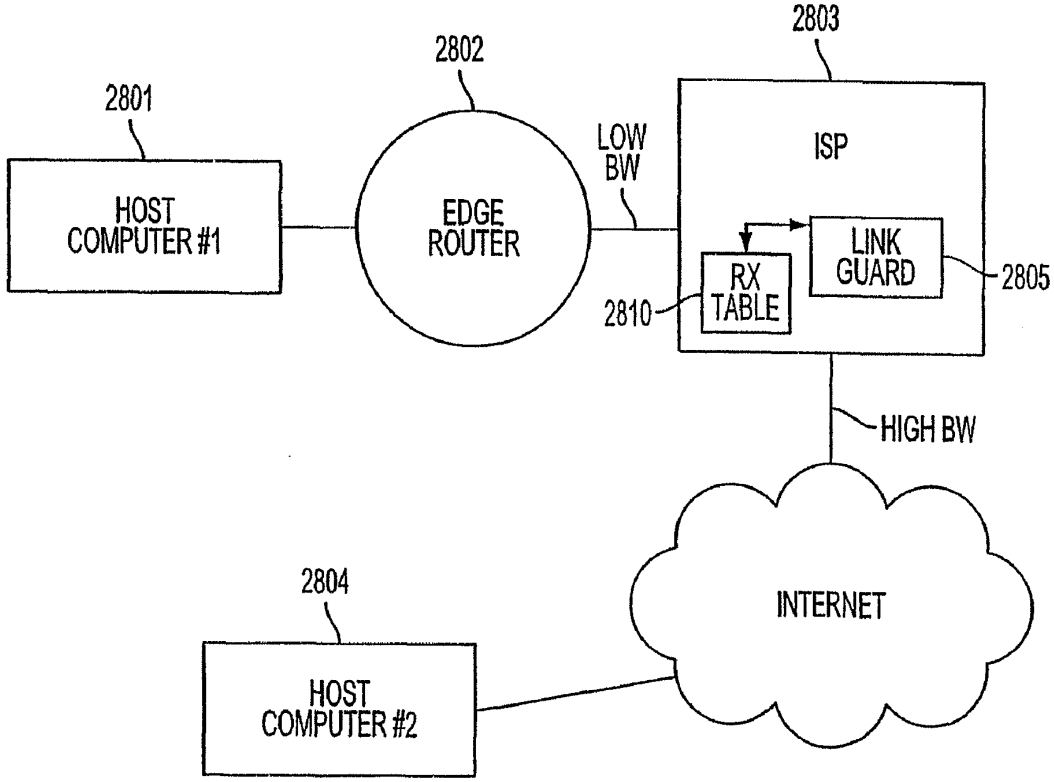

2801
HOST COMPUTER #1
2802
EDGE ROUTER
LOW BW
2803
ISP
RX TABLE
2810
LINK GUARD
2805
HIGH BW
INTERNET
2804
HOST COMPUTER #2

FIG. 28

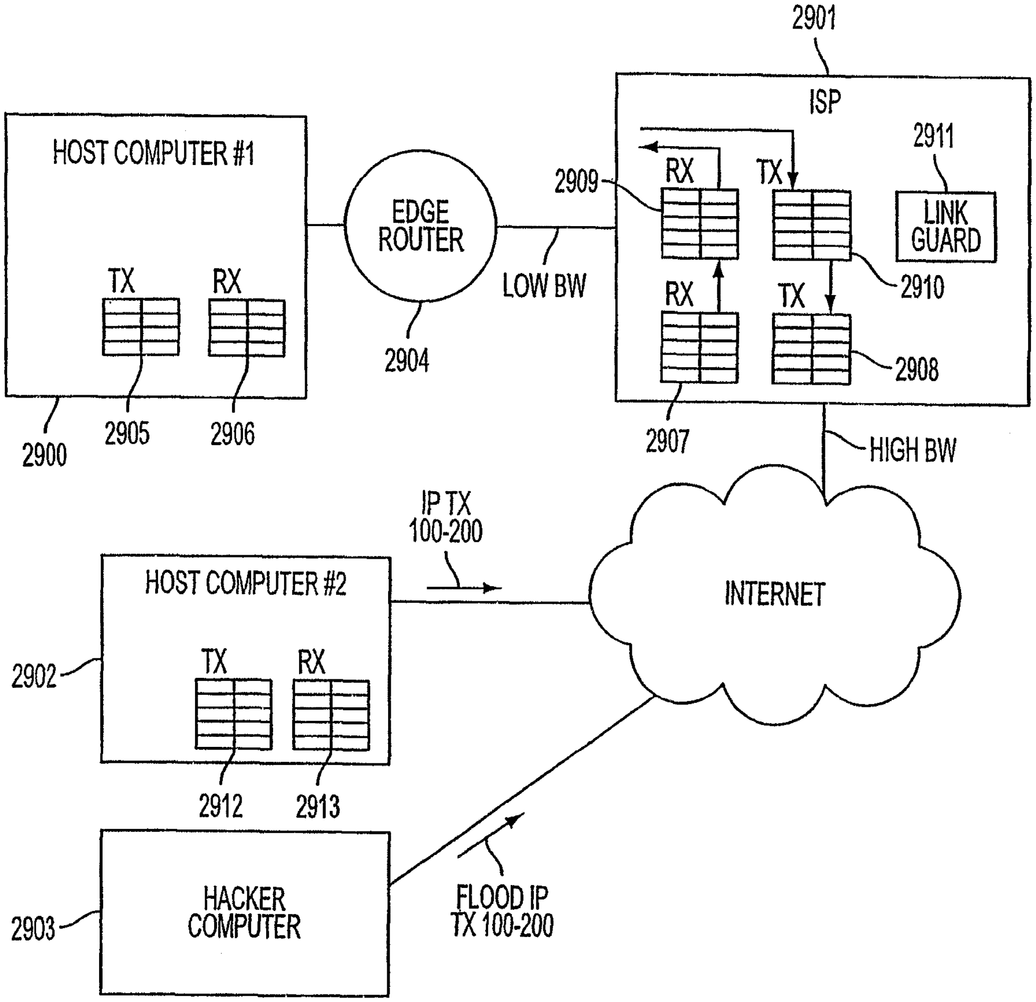

2901
ISP
2911
LINK GUARD
RX
TX
2909
2910
RX
TX
2908
2907
HOST COMPUTER #1
TX
RX
2905
2906
2900
EDGE ROUTER
2904
LOW BW
HIGH BW
INTERNET
IP TX 100-200
HOST COMPUTER #2
2902
TX
RX
2912
2913
HACKER COMPUTER
2903
FLOOD IP TX 100-200

FIG. 29

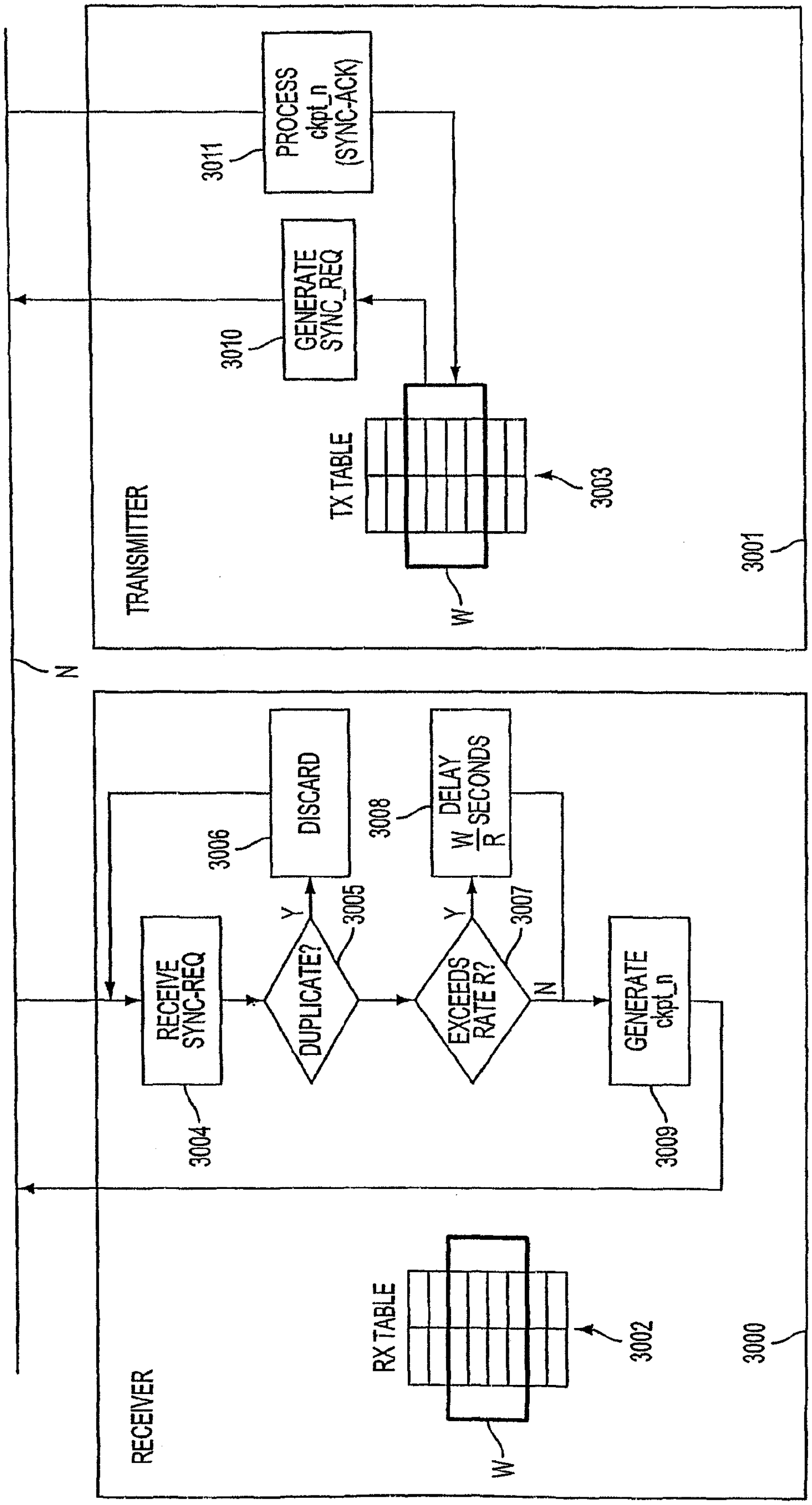

TRANSMITTER
TX TABLE
3003
W
3001
GENERATE SYNC_REQ
3010
PROCESS ckpt_n (SYNC-ACK)
3011
N
RECEIVER
RX TABLE
3002
W
3000
RECEIVE SYNC-REQ
3004
DUPLICATE?
3005
Y
DISCARD
3006
EXCEEDS RATE R?
3007
Y
DELAY $\frac{W}{R}$ SECONDS
3008
N
GENERATE ckpt_n
3009

FIG. 30

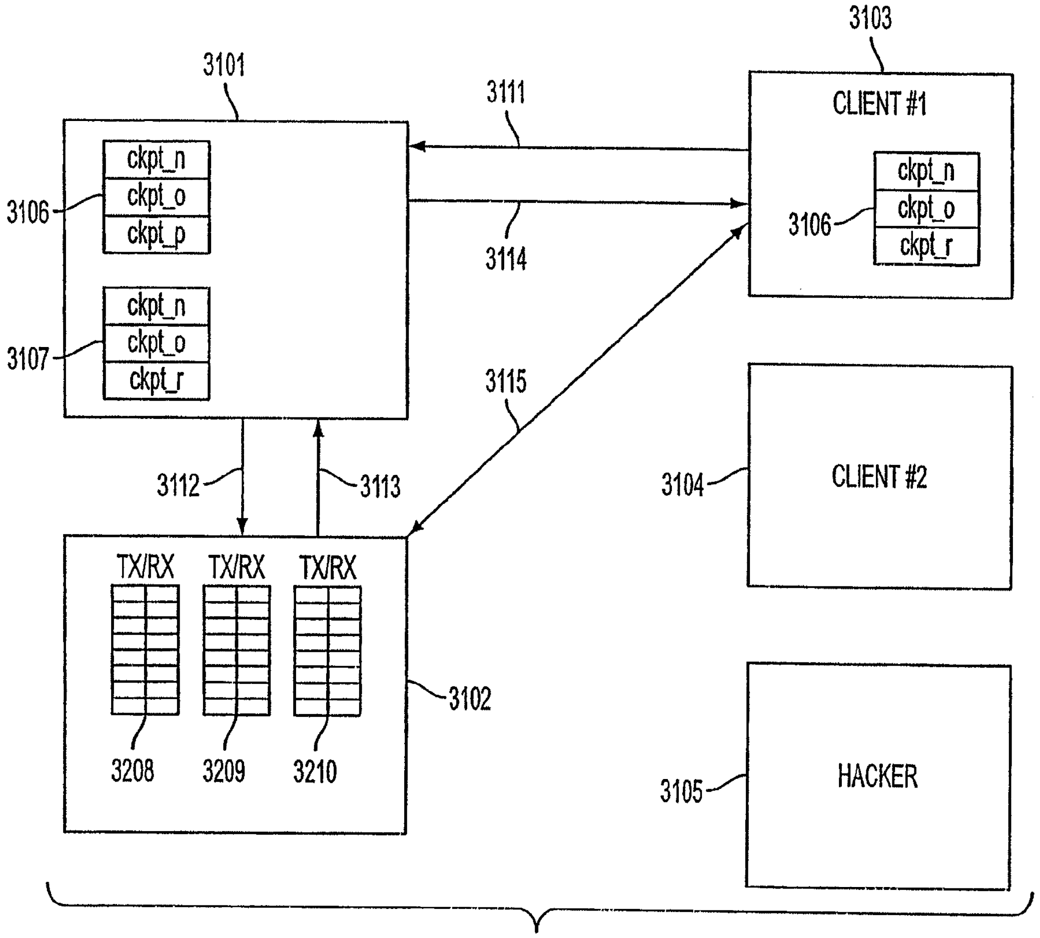

3101
3103
CLIENT #1
3111
ckpt_n
ckpt_o
ckpt_p
3106
3114
ckpt_n
ckpt_o
ckpt_r
3106
ckpt_n
ckpt_o
ckpt_r
3107
3115
3112
3113
CLIENT #2
3104
TX/RX
TX/RX
TX/RX
3102
3208
3209
3210
HACKER
3105

FIG. 31

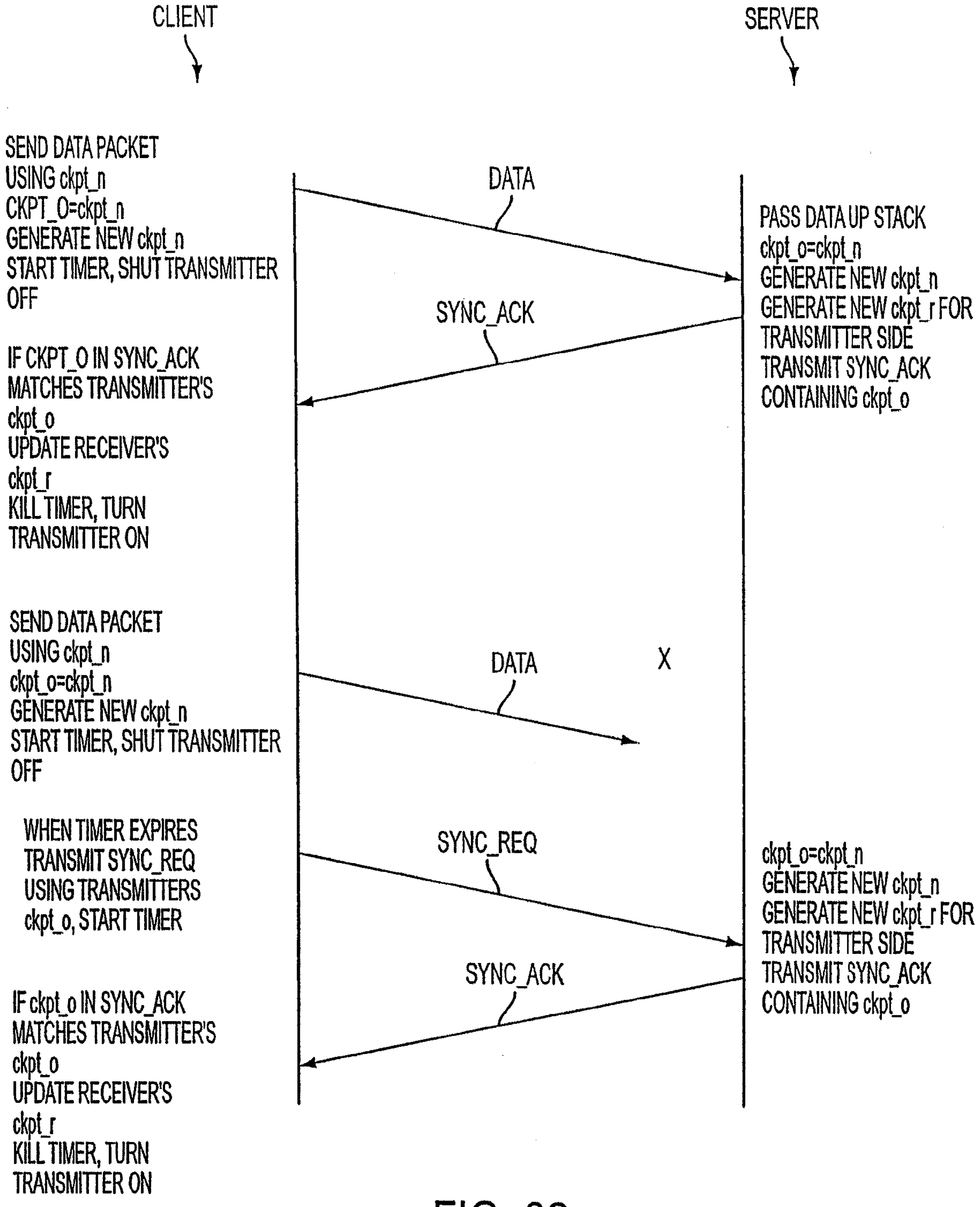

CLIENT
SERVER
SEND DATA PACKET
USING ckpt_n
CKPT_O=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
PASS DATA UP STACK
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING ckpt_n
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
X
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
ckpt_o, START TIMER
SYNC_REQ
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF ckpt_o IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

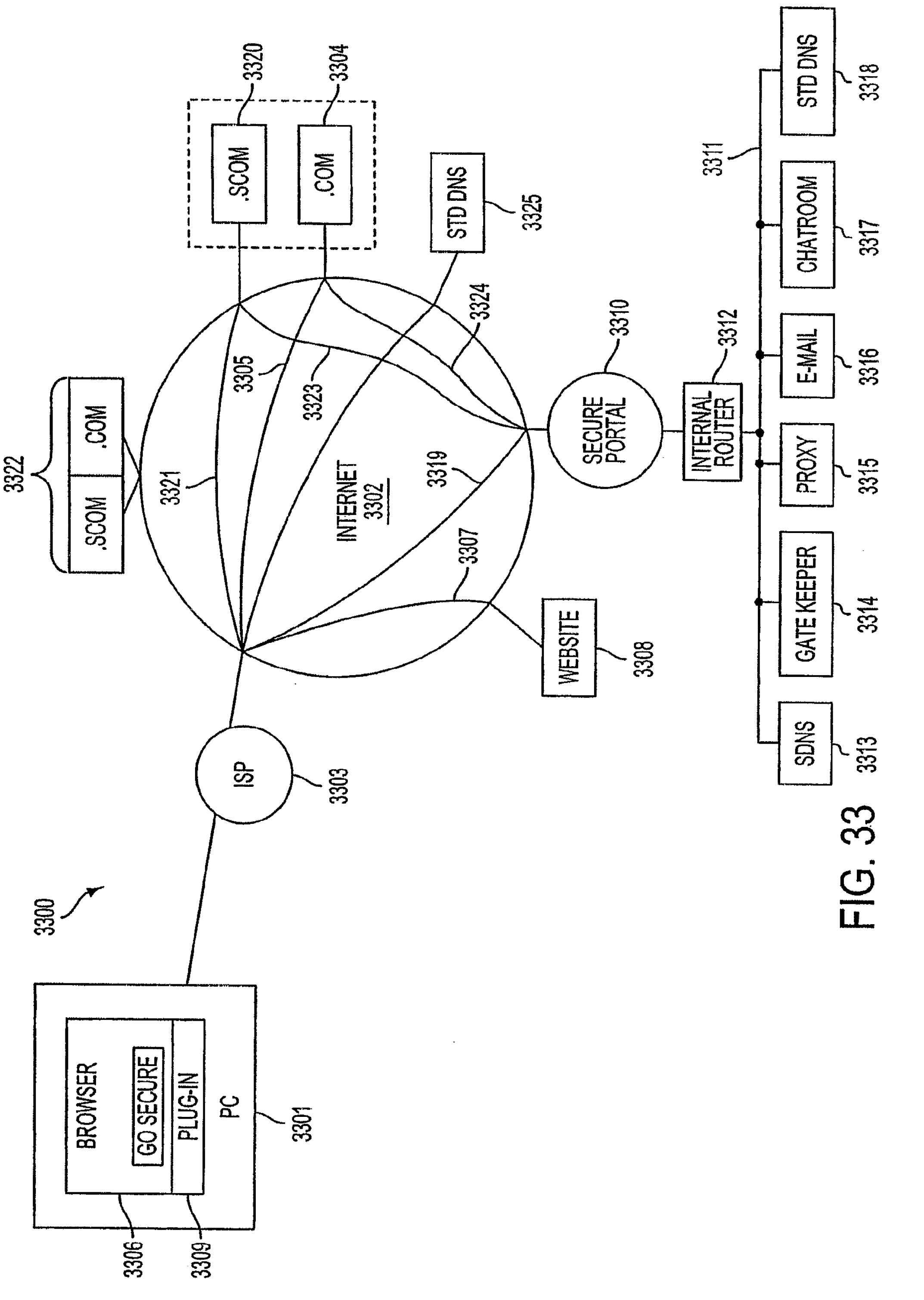

3300
PC
BROWSER
GO SECURE
PLUG-IN
3301
3306
3309
ISP
3303
INTERNET
3302
3305
3307
3319
3321
3323
3324
.SCOM
.COM
3322
.SCOM
.COM
3320
3304
STD DNS
3325
WEBSITE
3308
SECURE PORTAL
3310
INTERNAL ROUTER
3312
3311
SDNS
3313
GATE KEEPER
3314
PROXY
3315
E-MAIL
3316
CHATROOM
3317
STD DNS
3318

FIG. 33

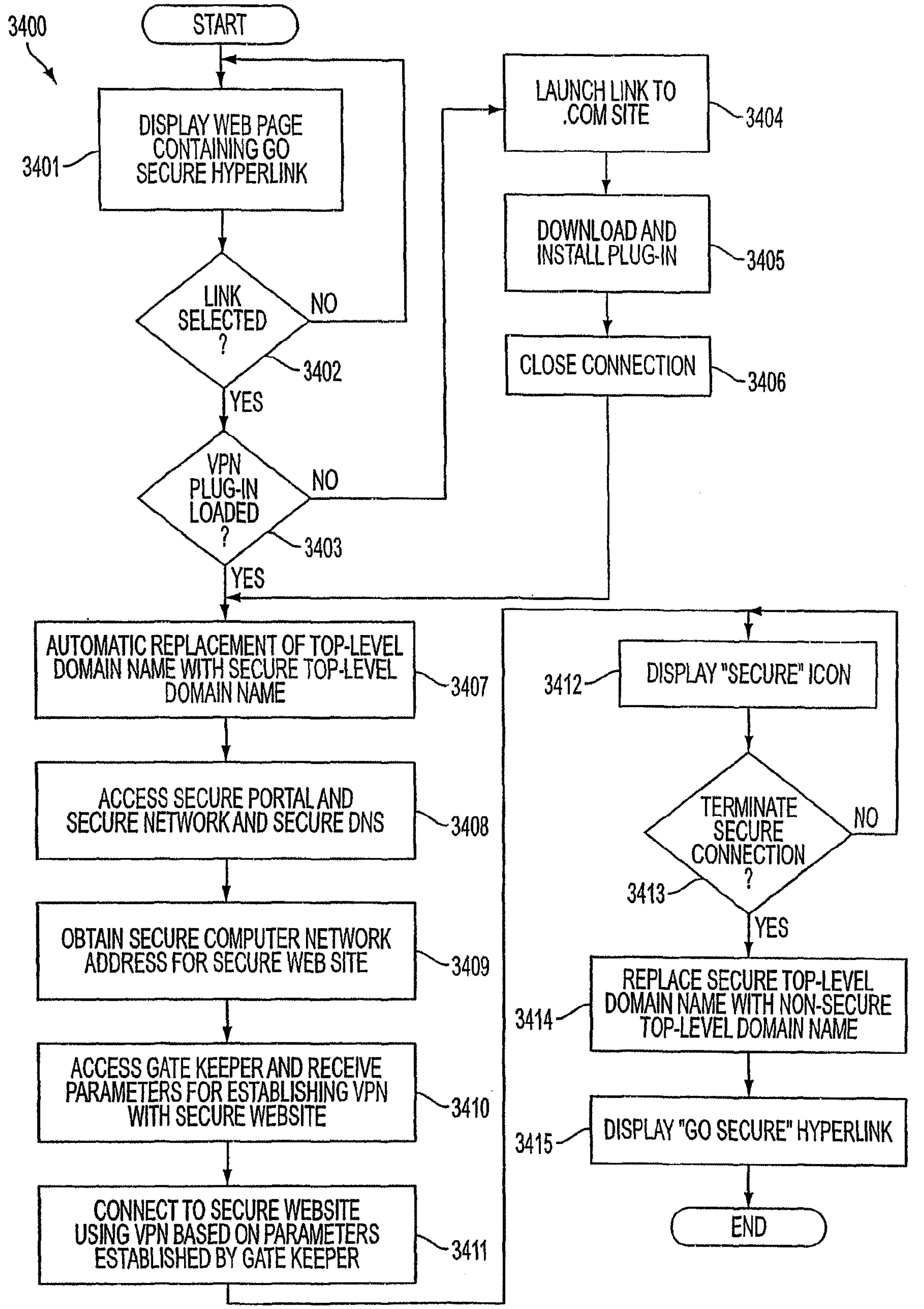

3400
START
DISPLAY WEB PAGE CONTAINING GO SECURE HYPERLINK
3401
LINK SELECTED ?
NO
3402
YES
VPN PLUG-IN LOADED ?
NO
3403
YES
LAUNCH LINK TO .COM SITE
3404
DOWNLOAD AND INSTALL PLUG-IN
3405
CLOSE CONNECTION
3406
AUTOMATIC REPLACEMENT OF TOP-LEVEL DOMAIN NAME WITH SECURE TOP-LEVEL DOMAIN NAME
3407
ACCESS SECURE PORTAL AND SECURE NETWORK AND SECURE DNS
3408
OBTAIN SECURE COMPUTER NETWORK ADDRESS FOR SECURE WEB SITE
3409
ACCESS GATE KEEPER AND RECEIVE PARAMETERS FOR ESTABLISHING VPN WITH SECURE WEBSITE
3410
CONNECT TO SECURE WEBSITE USING VPN BASED ON PARAMETERS ESTABLISHED BY GATE KEEPER
3411
3412
DISPLAY "SECURE" ICON
TERMINATE SECURE CONNECTION ?
NO
3413
YES
REPLACE SECURE TOP-LEVEL DOMAIN NAME WITH NON-SECURE TOP-LEVEL DOMAIN NAME
3414
3415
DISPLAY "GO SECURE" HYPERLINK
END

FIG. 34

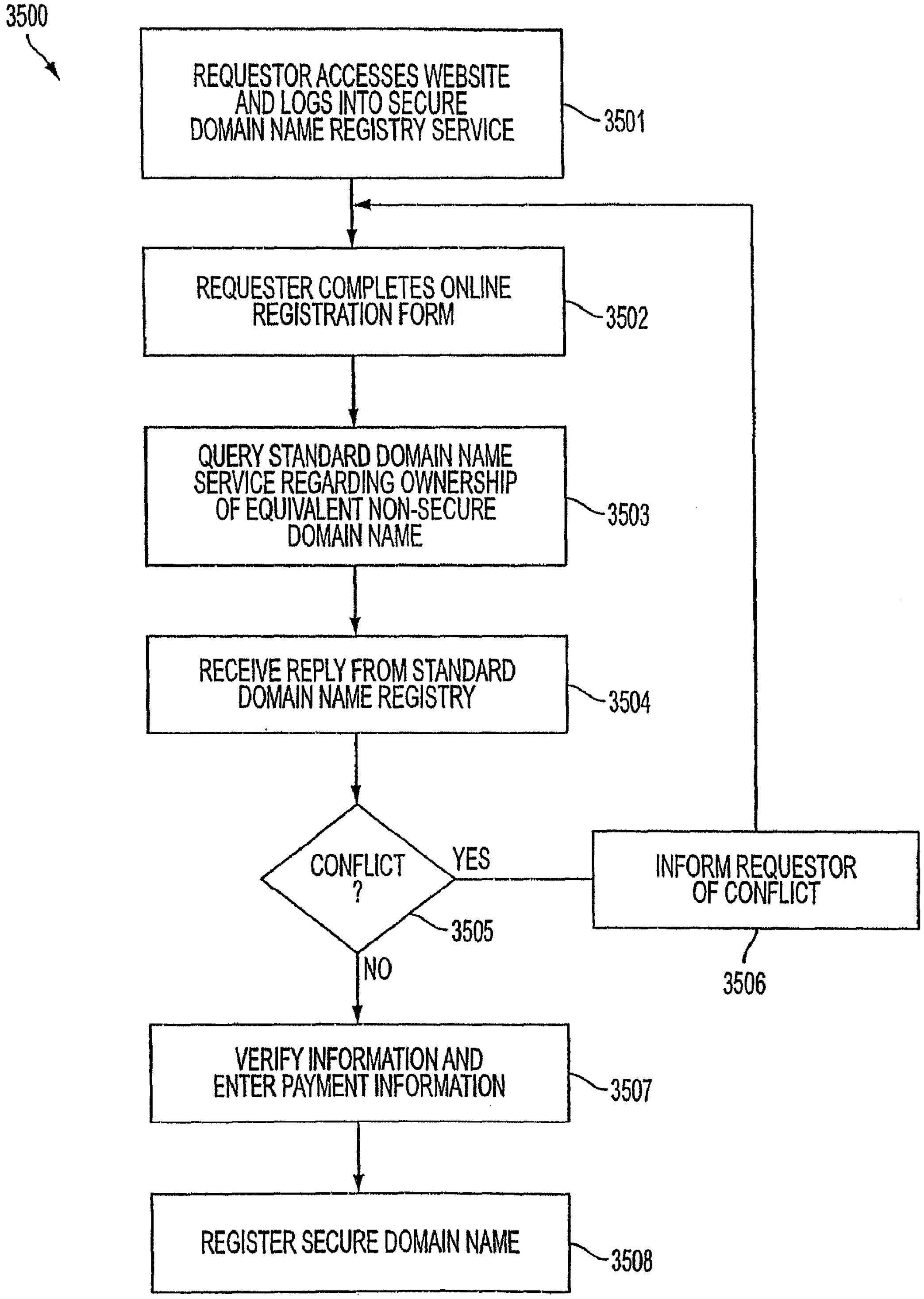

3500
REQUESTOR ACCESSES WEBSITE AND LOGS INTO SECURE DOMAIN NAME REGISTRY SERVICE
3501
REQUESTER COMPLETES ONLINE REGISTRATION FORM
3502
QUERY STANDARD DOMAIN NAME SERVICE REGARDING OWNERSHIP OF EQUIVALENT NON-SECURE DOMAIN NAME
3503
RECEIVE REPLY FROM STANDARD DOMAIN NAME REGISTRY
3504
CONFLICT ?
YES
INFORM REQUESTOR OF CONFLICT
3505
NO
3506
VERIFY INFORMATION AND ENTER PAYMENT INFORMATION
3507
REGISTER SECURE DOMAIN NAME
3508

FIG. 35

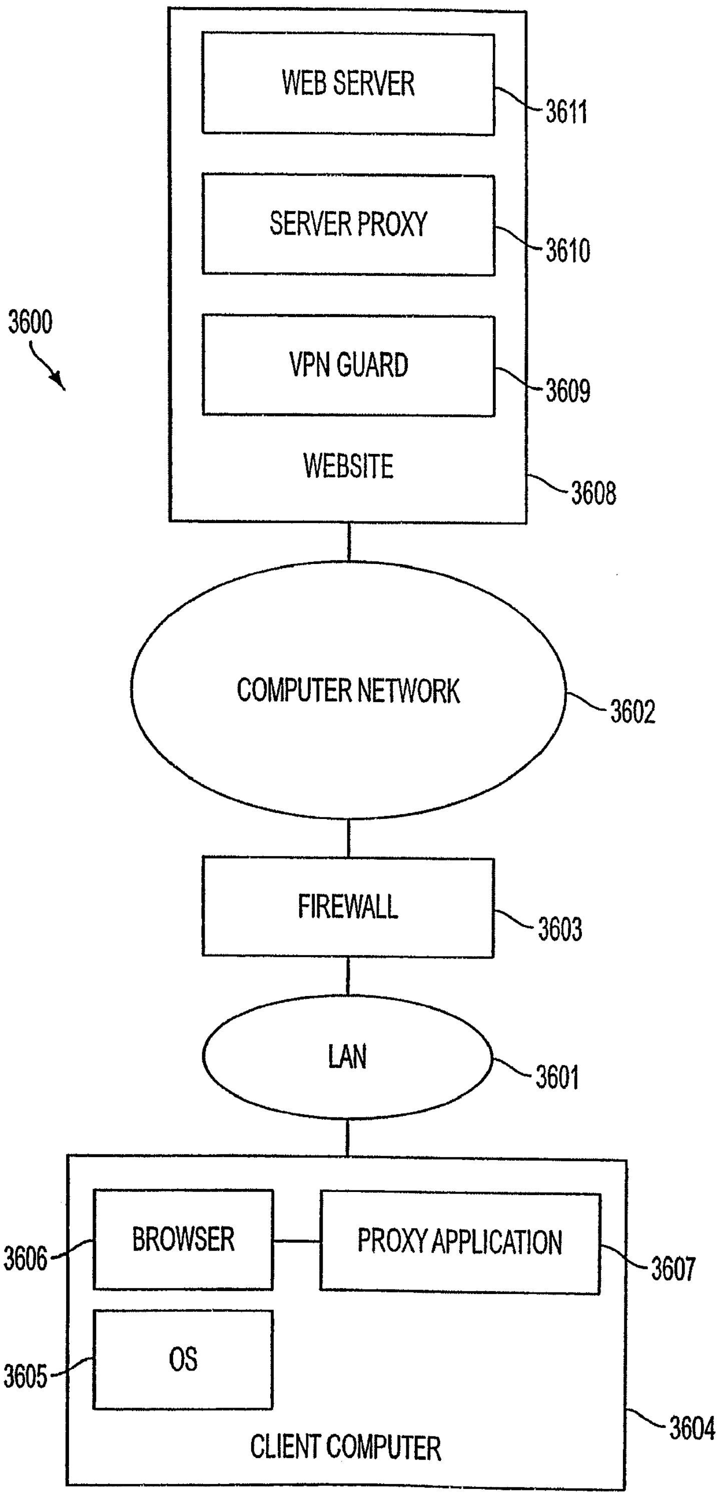

WEB SERVER
3611
SERVER PROXY
3610
3600
VPN GUARD
3609
WEBSITE
3608
COMPUTER NETWORK
3602
FIREWALL
3603
LAN
3601
BROWSER
3606
PROXY APPLICATION
3607
OS
3605
CLIENT COMPUTER
3604

FIG. 36

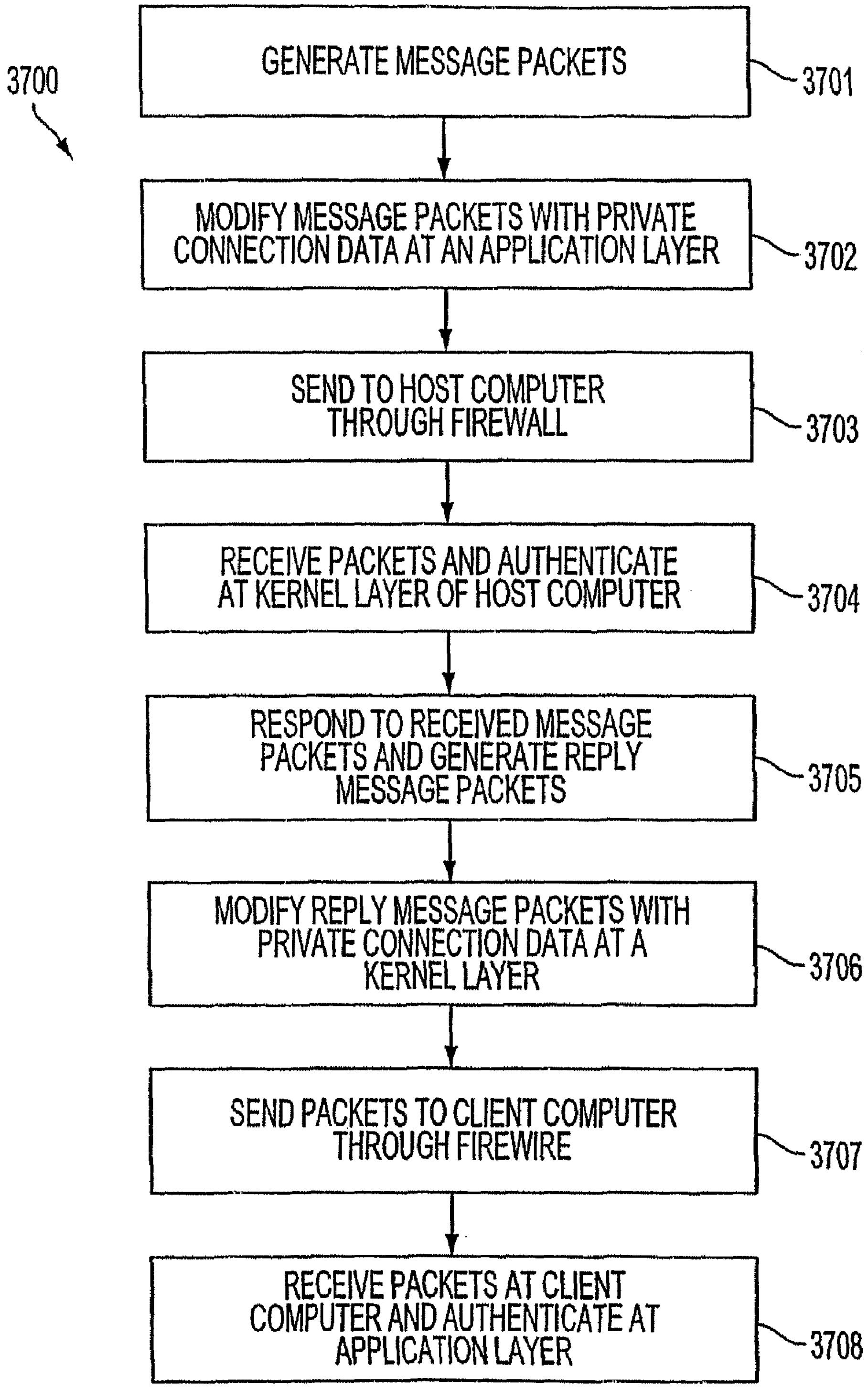

3700
GENERATE MESSAGE PACKETS
3701
MODIFY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT AN APPLICATION LAYER
3702
SEND TO HOST COMPUTER THROUGH FIREWALL
3703
RECEIVE PACKETS AND AUTHENTICATE AT KERNEL LAYER OF HOST COMPUTER
3704
RESPOND TO RECEIVED MESSAGE PACKETS AND GENERATE REPLY MESSAGE PACKETS
3705
MODIFY REPLY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT A KERNEL LAYER
3706
SEND PACKETS TO CLIENT COMPUTER THROUGH FIREWIRE
3707
RECEIVE PACKETS AT CLIENT COMPUTER AND AUTHENTICATE AT APPLICATION LAYER
3708

FIG. 37

AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from and is a continuation of a U.S. application Ser. No. 10/714,849, filed Nov. 18, 2003, now U.S. Pat. No. 7,418,504, which is a continuation of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000, now abandoned, which in turn is a continuation-in-part of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which in turn claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999, now U.S. Pat. No. 7,010,604, issued Mar. 07, 2006. The subject matter of U.S. application Ser. No. 09/429,643, now U.S. Pat. No. 7,010,604, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, now abandoned, and which is incorporated by reference herein.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. **1**. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client.

The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IP_T are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may he desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may he implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet transmitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply information packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet. Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12**A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12**B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. **20** shows a system employing distributed transmission paths.

FIG. **21** shows a plurality of link transmission tables that can be used to route packets in the system of FIG. **20**.

FIG. **22**A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. **22**B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. **23** shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. **24** shows an example using the system of FIG. **23**.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

FIG. 33 shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. 34 shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. 35 shows a flow diagram for registering a secure domain name according to the present invention.

FIG. 36 shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. 37 shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IPc. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq, to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IPT, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IPc is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100, 110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

I. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. 11, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP1 and IP2, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP3. Each frame header generally includes a source hardware address **1101** A and a destination hardware address **1101** B; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. **12A** shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. **12A**, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. **12A** and **12B** in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221** X, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window W1 maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12A** (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **121OX** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101** B in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12B** shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks, (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space Address

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be an prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. **14**).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. **15**. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send syne_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will he ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to he advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead Capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers X_1, X_2, X_3 . . . Xk starting with seed X_0 using a recurrence

$$X_i=(aX_{i-1}+b)\text{mod } c, \qquad (1)$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1))\text{mod } c \qquad (2)$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1)\text{mod } c. \qquad (3)$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1)\text{mod } c=((a^i \text{ mod}((a-1)c)(X_0(a-1)+b)-b)/(a-1))\text{mod } c \qquad (4).$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b)$ mod c, b as b mod c and compute a^i mod((a−1)c) (this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the j^{th} checkpoint, as X_0 and n as i, a node can store a^n mod((a−1)c) once per LCR and set

$$X_{j+1}{}^w=X_{n(j+1)}=((a^n \text{ mod}((a-1)c)(X_j^w(a-1)+b)-b)/(a-1))\text{mod } c, \qquad (5)$$

to generate the random number for the $j+1^{th}$ synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4)\text{mod } 15. \qquad (6)$$

If one sets X_0=1, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, c*(a−1)=15*30=450 and a^n mod ((a−1)c)=31^3 mod(15*30)=29791 mod(450)=91. Equation (5) becomes:

$$((91(X_i30+4)-4)/30)\text{mod } 15 \qquad (7).$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91\ (X_i30 + 4) - 4$	$((91\ (X_i30 + 4) - 4)/30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (–16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B—trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^n-1). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector (s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. **16**). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses (I≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧1). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. **17** depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. **18** when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. **19** shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE—OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is predetermined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. Continuation-in-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. **22A** shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. **22A**. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22A** the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. **22B** shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P'=\alpha\times\mathrm{MIN}+(1-\alpha)\times P \qquad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P'=\beta\times S+(1-\beta)\times P \qquad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. **25**. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**.

When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project (RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **04**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **03** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **01**, a DNS look-up request is received for a target host. In step **02**, a check is made to determine whether access to a secure host was requested. If not, then in step **03** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **02**, if access to a secure host was requested, then in step **04** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **05**). If the user has sufficient security privileges, then in step **06** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses **100** to **200** for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range **100** to **200**, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC_ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets, A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC_REQ every T1 seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.
2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of M×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×W/R seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer T1 noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.
2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e., user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPTO associated with the timer, it starts timer T1 noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.
5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

F. One-Click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. **33** shows a system block diagram **3300** of a computer network in which the one-click secure communication method of the present invention is suitable. In FIG. **33**, a computer terminal or client computer **3301**, such as a personal computer (PC), is connected to a computer network **3302**, such as the Internet, through an ISP **3303**. Alternatively, computer **3301** can be connected to computer network **3302** through an edge router. Computer **3301** includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer **3301** can communicate conventionally with another computer **3304** connected to computer network **3302** over a communication link **3305** using a browser **3306** that is installed and operates on computer **3301** in a well-known manner.

Computer **3304** can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network **3302** is the Internet, computer **3304** typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. **34** shows a flow diagram **3400** for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step **3401**, computer **3301** is connected to server computer **3304** over a non-VPN communication link **3305**. Web browser **3306** displays a web page associated with server **3304** in a well-known manner. According to one variation of the invention, the display of computer **3301** contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network **3302** between terminal **3301** and server **3304**. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer **3304**, thereby indicating that the entity providing server **3304** also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer **3301** is informed that the current communication link between computer **3301** and server computer **3304** is a non-secure, non-VPN communication link. At step **3402**, it is determined whether a user of computer **3301** has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step **3402**, it is determined that the user has selected the "go secure" hyperlink, flow continues to step **3403** where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer **3301**. Alternatively, a user can enter a command into computer **3301** to "go secure."

If, at step **3403**, the object determines that the software module has been installed, flow continues to step **3407**. If, at step **3403**, the object determines that the software module has not been installed, flow continues to step **3404** where a non-VPN communication link **3307** is launched between computer **3301** and a website **3308** over computer network **3302** in a well-known manner. Website **3308** is accessible by all computer terminals connected to computer network **3302** through a non-VPN communication link. Once connected to website **3308**, a software module for establishing a secure communication link over computer network **3302** can be downloaded and installed. Flow continues to step **3405** where, after computer **3301** connects to website **3308**, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal **3301** as software module **3309**. At step **3405**, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network **3302**. At step **3406**, the communication link between computer **3301** and website **3308** is then terminated in a well-known manner.

By clicking on the "go secure" hyperlink, a user at computer **3301** has enabled a secure communication mode of communication between computer **3301** and server computer **3304**. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer **3301** and server computer **3304** are performed transparently to a user at computer **3301**.

At step **3407**, a secure VPN communications mode of operation has been enabled and software module **3309** begins to establish a VPN communication link. In one embodiment, software module **3309** automatically replaces the top-level domain name for server **3304** within browser **3406** with a secure top-level domain name for server computer **3304**. For example, if the top-level domain name for server **3304** is .com, software module **3309** replaces the .com top-level domain name with a .scom top-level domain name, where the "s" stands for secure. Alternatively, software module **3409** can replace the top-level domain name of server **3304** with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module **3409** contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module **3309** accesses a secure portal **3310** that interfaces a secure network **3311** to computer network **3302**. Secure network **3311** includes an internal router **3312**, a secure domain name service (SDNS) **3313**, a VPN gatekeeper **3314** and a secure proxy **3315**. The secure network can include other network services, such as e-mail **3316**, a plurality of chatrooms (of which only one chatroom **3317** is shown), and a standard domain name service (STD DNS) **3318**. Of course, secure network **3311** can include other resources and services that are not shown in FIG. **33**.

When software module **3309** replaces the standard top-level domain name for server **3304** with the secure top-level domain name, software module **3309** sends a query to SDNS **3313** at step **3408** through secure portal **3310** preferably using an administrative VPN communication link **3319**. In this configuration, secure portal **3310** can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal **3310** authenticates the query from software module **3309** based on the particular information hopping technique used for VPN communication link **3319**.

SDNS **3313** contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS **3313** stores a computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS **3313** so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS **3313** for the secure computer network address for the securities trading website, SDNS **3313** determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step **3409**, SDNS **3313** accesses VPN gatekeeper **3314** for establishing a VPN communication link between software module **3309** and secure server **3320**. Server **3320** can only be accessed through a VPN communication link. VPN gatekeeper **3314** provisions computer **3301** and secure web server computer **3320**, or a secure edge router for server computer **3320**, thereby creating the VPN. Secure server computer **3320** can be a separate server computer from server computer **3304**, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer **3322**. Returning to FIG. **34**, in step **3410**, SDNS **3313** returns a secure URL to software module **3309** for the .scom server address for a secure server **3320** corresponding to server **3304**.

Alternatively, SDNS **3313** can be accessed through secure portal **3310** "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal **3310** preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS **3319**. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS **3313** can be in the clear, and SDNS **3313** and gatekeeper **3314** can operate to establish a VPN communication link to the querying computer for sending the reply.

At step **3411**, software module **3309** accesses secure server **3320** through VPN communication link **3321** based on the VPN resources allocated by VPN gatekeeper **3314**. At step **3412**, web browser **3306** displays a secure icon indicating that the current communication link to server **3320** is a secure VPN communication link. Further communication between computers **3301** and **3320** occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link **3321** is terminated at step **3413**, flow continues to step **3414** where software module **3309** automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server **3304**. Browser **3306** accesses a standard DNS **3325** for obtaining the non-secure URL for server **3304**. Browser **3306** then connects to server **3304** in a well-known manner. At step **3415**, browser **3306** displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal **3301** and server **3304**. By again displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module **3309** is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network **3302** are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module **3309** transparently accesses SDNS **3313** for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to he effected.

Additionally, a user at computer **3301** can optionally select a secure communication link through proxy computer **3315**. Accordingly, computer **3301** can establish a VPN communication link **3323** with secure server computer **3320** through proxy computer **3315**. Alternatively, computer **3301** can establish a non-VPN communication link **3324** to a non-secure website, such as non-secure server computer **3304**.

FIG. **35** shows a flow diagram **3500** for registering a secure domain name according to the present invention. At step **3501**, a requester accesses website **3308** and logs into a secure domain name registry service that is available through website **3308**. At step **3502**, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step **3503**, the secure domain name registry service at website **3308** queries a non-secure domain name server database, such as standard DNS **3322**, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step **3504**, the secure domain name registry service at website **3308** receives a reply from standard DNS **3322** and at step **3505** determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step **3507**, otherwise flow continues to step **3506** where the requestor is informed of the conflicting ownership information. Flow returns to step **3502**.

When there is no conflicting ownership information at step **3505**, the secure domain name registry service (website **3308**) informs the requestor that there is no conflicting ownership information and prompts the requestor to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step **3508** where the newly registered secure domain name sent to SDNS **3313** over communication link **3326**.

If, at step **3505**, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requestor for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS **3325** in a well-known manner. Flow then continues to step **3508**.

G. Tunneling Secure Address Hopping Protocol Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. 36 shows a system block diagram of a computer network 3600 in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. 37 shows a flow diagram 3700 for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. 36 a local area network (LAN) 3601 is connected to another computer network 3602, such as the Internet, through a firewall arrangement 3603. Firewall arrangement operates in a well-known manner to interface LAN 3601 to computer network 3602 and to protect LAN 3601 from attacks initiated outside of LAN 3601.

A client computer 3604 is connected to LAN 3601 in a well-known manner. Client computer 3604 includes an operating system 3605 and a web browser 3606. Operating system 3605 provides kernel mode functions for operating client computer 3604. Browser 3606 is an application program for accessing computer network resources connected to LAN 3601 and computer network 3602 in a well-known manner. According to the present invention, a proxy application 3607 is also stored on client computer 3604 and operates at an application layer in conjunction with browser 3606. Proxy application 3607 operates at the application layer within client computer 3604 and when enabled, modifies unprotected, unencrypted message packets generated by browser 3606 by inserting data into the message packets that are used for forming a virtual private connection between client computer 3604 and a server computer connected to LAN 3601 or computer network 3602. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application 3607 is conveniently installed and uninstalled by a user because proxy application 3607 operates at the application layer within client computer 3604. On installation, proxy application 3607 preferably configures browser 3606 to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer 3604 and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application 3606 can be selected and enabled through, for example, an option provided by browser 3606. Additionally, proxy application 3607 can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer 3604 and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. 37, at step 3701, unprotected and unencrypted message packets are generated by browser 3606. At step 3702, proxy application 3607 modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer 3604 and a destination server computer into the payload portion. At step, 3703, the modified message packets are sent from client computer 3604 to, for example, website (server computer) 3608 over computer network 3602.

Website 3608 includes a VPN guard portion 3609, a server proxy portion 3610 and a web server portion 3611. VPN guard portion 3609 is embedded within the kernel layer of the operating system of website 3608 so that large bandwidth attacks on website 3608 are rapidly rejected. When client computer 3604 initiates an authenticated connection to website 3608, VPN guard portion 3609 is keyed with the hopping sequence contained in the message packets from client computer 3604, thereby performing a strong authentication of the client packet streams entering website 3608 at step 3704. VPN guard portion 3609 can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion 3609 can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion 3609 would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

Server proxy portion 3610 also operates at the kernel layer within website 3608 and catches incoming message packets from client computer 3604 at the VPN level. At step 3705, server proxy portion 3610 authenticates the message packets at the kernel level within host computer 3604 using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion 3611 as normal TCP web transactions.

At step 3705, web server portion 3611 responds to message packets received from client computer 3604 in accordance with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion **3611** generates message packets corresponding to the requested webpage. At step **3706**, the reply message packets pass through server proxy portion **3610**, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer **3608** and client computer **3604** over computer network **3602**. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion **3610** operates at the kernel layer within host computer **3608** to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer **3608** to client computer **3604** conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step **3707**, the modified packets are sent from host computer **3608** over computer network **3602** and pass through firewall **3603**. Once through firewall **3603**, the modified packets are directed to client computer **3604** over LAN **3601** and are received at step **3708** by proxy application **3607** at the application layer within client computer **3604**. Proxy application **3607** operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising:

a domain name service system configured and arranged to be connected to a communication network, store a plurality of domain names and corresponding network addresses, receive a query for a network address, and indicate in response to the query whether the domain name service system supports establishing a secure communication link.

2. The system of claim **1**, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim **2**, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim **3**, wherein the non-standard top-level domain name is one of .scom, .sorg, .snet, .sgov, .sedu, .smil and .sint.

5. The system of claim **2**, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

6. The system of claim **1**, wherein the communication network includes the Internet.

7. The system of claim **1**, wherein the domain name service system comprises an edge router.

8. The system of claim **1**, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim **8**, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

10. The system of claim **8**, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim **8**, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim **8**, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim **8**, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim **1**, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim **1**, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim **1**, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim **1**, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim **1**, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim **1**, wherein the domain name service system comprises a server.

20. The system of claim **19**, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim **1**, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim **1**, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.

23. The system of claim **1**, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim **1**, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim **1**, wherein at least one of the plurality of domain names comprises a secure name.

26. The system of claim **1**, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim **1**, wherein the secure communication link uses encryption.

29. The system of claim **1**, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim **29**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim **30**, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim **29**, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim **33**, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim **1**, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A non-transitory machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for: connecting the domain name service system to a communication network; storing a plurality of domain names and corresponding network addresses; receiving a query for a network address; and indicating in response to the query whether the domain name service system supports establishing a secure communication link.

37. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for responding to the query for the network address.

39. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for providing, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for indicating that the domain name service system supports the establishment of a secure communication link.

42. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The non-transitory machine-readable medium of claim **36**, wherein the code resides on a server.

44. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The non-transitory machine-readable medium of claim **36**, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for authenticating the query for the network address.

48. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes a secure name.

50. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The non-transitory machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The non-transitory machine-readable medium of claim **36**, wherein the secure communication link uses encryption.

53. The non-transitory machine-readable medium of claim **36**, wherein the secure communication link is capable of supporting a plurality of services.

54. The non-transitory machine-readable medium of claim **53**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The non-transitory machine-readable medium of claim **54**, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

56. The non-transitory machine-readable medium of claim **53**, wherein the plurality of services comprises audio, video, or a combination thereof.

57. The non-transitory machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The non-transitory machine-readable medium of claim **57**, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

59. The non-transitory machine-readable medium of claim **36**, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name is response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

connecting a domain name service system to a communication network;

storing a plurality of domain names and corresponding network addresses; and

upon receiving a query for a network address for communication, indicating whether the domain name service system supports establishing a secure communication link.

* * * * *